UNIVERZITET U BEOGRADU
Elektrotehnički fakultet
Katedra za računarsku tehniku i informatiku

Testiranje Softvera

Vežbe - Strategije crne kutije

Profesor:
dr Dragan Bojić

Asistent:
dipl. ing. Dražen Drašković

Beograd, 2012.

Teorija (Osnove):

Testiranje crnom kutijom (Black Box Strategy), kao što sam naziv kaže, je strategija koja zahteva da se program posmatra kao zatvoreni sistem (crna kutija) kod kojeg se analizira odziv na određenu pobudu, tj. reakcija na zadate ulazne podatke. Ova strategija, za razliku od strategija bele kutije, ne zahteva poznavanje unutrašnjeg dizajna koda i analizu izvornog koda, već samo osobine definisane specifikacijom softverske aplikacije koja se testira. Zato je ova strategija potpuno fokusirana na funkcionalnostima rada aplikacije.

Specifikacija aplikacije mora da ispunjava tri uslova: da bude tačna, razumljiva i potpuna. Ulazni podaci za testiranje aplikacije definišu se tako da povećaju verovatnoću nalaženja greške i da smanje veličinu skupa testova, odnosno broja test primera.

Dve najpoznatije metode koje pripadaju strategiji crne kutije su:

- Metoda klasa ekvivalencije
- Metoda graničnih vrednosti

Ovo testiranje se često naziva i funkcionalnim testiranjem (testiranjem ponašanja), neprozirnim testiranjem ili zatvorenim testiranjem.

Različiti tipovi testiranja spadaju u strategiju crne kutije:

- funkcionalno testiranje (functional testing)
- testiranje pod pritiskom (stress testing)
- testiranje oporavka (recovery testing)
- testiranje opsega (volume testing)
- testiranje prihvatljivosti od strane korisnika (user acceptance testing)
- testiranje sistema (system testing)
- testiranje opterećenja (load testing)
- testiranje upotrebljivosti (usability testing)
- istraživačko testiranje (exploratory testing)
- ad-hoc testiranje (ad-hoc testing)
- alfa testiranje (alpha testing)
- beta testiranje (beta testing)

METOD DELJENJA NA KLASE EKVIVALENCIJE

(eng. Equivalence Class Partitioning)

Teorija (Osnove):

- Ideja je da se skup svih mogućih ulaznih podataka izdeli na podskupove (klase), pri čemu u istu klasu ulaze oni ulazni podaci koji daju iste (slične) rezultate, na primer, otkrivaju istu grešku.
- U idealnom slučaju podskupovi su međusobno disjunktni i pokrivaju ceo skup ulaza (relacija "sličnosti" ulaza je relacija ekvivalencije).

- U cilju identifikacije klasa, posmatraju se svi uslovi koji proizilaze iz specifikacije.

  Za svaki uslov se posmatraju dva grupe klasa prema zadovoljenosti uslova:
  - legalne klase obuhvataju ispravne situacije (ulazne podatke).
  - nelegalne klase obuhvataju sve ostale situacije (ulazne podatke).

- Uputstva za identifikaciju klasa:
  1. Ako neki uslov podrazumeva opseg vrednosti nekog ulaznog podatka, postoji jedna legalna klasa (unutar opsega) i dve nelegalne klase (ispod i iznad opsega).
  2. Ako neki uslov podrazumeva broj vrednosti nekog ulaznog podatka, postoji jedna legalna klasa (dozvoljen broj pojavljivanja) i dve nelegalne klase (ni jedno pojavljivanje i suviše pojavljivanja).
  3. Ako neki uslov podrazumeva skup ulaznih vrednosti od kojih se svaka na poseban način obrađuje u programu, postoji za svaku vrednost po jedna legalna klasa i jedna zajednička nelegalna klasa.
  4. Ako neki uslov podrazumeva situaciju obaveznosti ("mora biti slovo"), postoji jedna legalna klasa ("jeste slovo") i jedna nelegalna klasa ("nije slovo").
  5. U slučaju da program ne tretira jednako sve elemente neke klase ekvivalencije, nju treba podeliti na više manjih klasa.
- Prednost ovog pristupa je smanjenje vremena potrebnog za testiranje softvera, zbog manjeg broja ispitanih test slučajeva - vrednosti unutar jedne particije se smatraju „ekvivalentnim", pa je dovoljno izabrati jedan test slučaj iz svake particije da bi se proverilo ponašanje programa. Ako bi se ispitali veći broj ili čak sve test slučajeve u istoj particiji, oni ne bi našli nove greške u programu.

Zadatak 1:

Metodom deljenja na klase ekvivalencije testirati deo FORTRAN prevodioca koji ispituje sintaksnu ispravnost naredbe

DIMENSION ad [,ad]...

gde je ad oznaka polja u formatu

ad ::= n (d [,d]..)

gde je n simboličko ime polja, niz od 1 do 6 slova i cifara, prvi znak uvek slovo, a d dimenzija (može ih biti najmanje 1 a najviše 7) u obliku

d ::= [lb:] ub

gde je lb donja granica (ako nije navedena, podrazumeva se 1), a ub gornja granica. Granice mogu biti celobrojne konstante u intervalu od -65536 do 65535 ili imena celobrojnih promenljivih. Naredba DIMENSION može se pisati u više redova, pri čemu se u svakom sledećem redu osim prvog u 6 koloni piše znak različit od razmaka.

Rešenje:

- Prvi korak je u identifikaciji uslova za ulazne podatke (u ovom slučaju to je DIMENSION naredba) i odgovarajućih klasa ekvivalencije.
- Drugi korak je u formiranju skupa test-primera koji pokrivaju sve legalne klase ekvivalencije (jedan test primer sme da pokrije veći broj klasa)
- Poslednji korak je u formiranju posebnog test-primera za svaku od nelegalnih klasa.

Uslov	legalne klase	Nelegalne klase
broj polja	1 (1) , >1 (2)	ni jedno (3)
dužina imena	1 - 6 (4)	0 (5) , >6 (6)
ime polja sadrži	slova (7) , brojeve (8)	nešto drugo (9)
ime poč, slovom	da (10)	ne (11)
broj dimenzija	1 (12) , 2-7 (13)	0 (14) , >7 (15)
Gornja granica	konstanta (16) int. var (17)	element polja (18) nešto drugo (19)
Ime int.v sadrži	slova (20) brojeve (21)	nešto drugo (22)
int.v. poč.slovom	da (23)	ne (24)
Konstanta je	-65536 – 65535 (25)	<-65536 (26) >65535 (27)
Donja granica je specificirana	da (28) , ne (29)	
Gornja granica je u odnosu na donju	veća (30), jednaka (31)	manja (32)
Specificirana donja granica je	<0 (33) , =0 (34) >0 (35)	
Donja granica je	konstanta (36) int. var (37)	element polja (38) nešto drugo (39)
U više redova	da (40) , ne (41)	

Test primeri:

1) DIMENSION A(2)
pokriva: 1, 4, 7, 10, 12, 16, 25, 29, 30, 41

2) DIMENSION A12345(1,9,J4XXXX,65534,ABC:KLM,
1 100),BBB(-65535:100,0:1000,10:10,1:65534)
pokriva ostale legalne slučajeve.

Nelegalni slučajevi:

3 DIMENSION	19 DIMENSION B(4,,7)
5 DIMENSION (10)	22 DIMENSION C(.,10)
6 DIMENSION A234567(20)	24 DIMENSION C(10,1J)
9 DIMENSION A.1(2)	26 DIMENSION D(-65537:1)
11 DIMENSION 1A(10)	27 DIMENSION D(65536)
14 DIMENSION B	32 DIMENSION D(4:3)
15 DIMENSION B(4,4,4,4,4,4,4,4)	38 DIMENSION D(A(2):4)
18 DIMENSION B(4,A(2))	39 DIMENSION D(.:4)

Zadatak 2: Generator Ocena

Komponenta prihvata broj poena na ispitu (vrednost do 75) i broj poena osvojen na domaćem (vrednost do 25), na onovu kojih generiše ocenu na predmetu u rangu od ‘A’ do ‘D’. Ocena se računa na osnovu ukupnog broja poena koji se dobija kao suma ocene na ispit i domaćem:

veće ili jednako 70	- ‘A’
veće ili jednako 50, ali manje od 70	- ‘B’
veće ili jednako 30, ali manje od 50	- ‘C’
manje od 30	- ‘D’

Ukoliko je ukupan broj poena izvan očekivanog opsega, (‘FM’) se generiše kao izlaz. Svi ulazi su celobrojne vrednosti.

Najpre se odrede klase ekvivalencije a zatim se generišu odgovarajući test primeri.
Klase ekvivalencije moguće je definisati i za ulaze i za izlaze komponente! Takođe i validni i nevalidni ulazi i izlazi se razmatraju.

Rešenje:

Inicijalno se mogu identifikovati particije za dva ulaza. Validne particije možemo opisati:

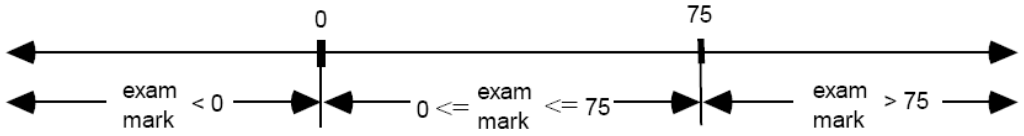
0 <= broj poena na ispitu <= 75
0 <= broj poena na domaćem <= 25

Najočiglednije nevalidne particije, ulaza možemo opisati:

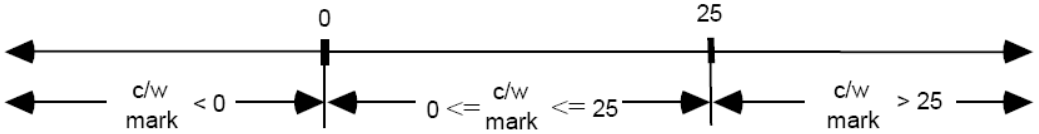
broj poena na ispitu > 75
broj poena na ispitu < 0
broj poena na domaćem > 25
broj poena na domaćem < 0

Identifikovane oblasti možemo prikazati grafički.

Što se tiče ulazne ocene sa ispita, grafički, oblasti može prikazati na sledeći način:



Slično, za ulaznu vrednost, broj poena na domaćem, imamo:



Manje očigledne ulazne klase ekvivalencije mogu uključiti bilo koje ulaze koje nisu uključene u prethodne klase ekvivalencije. Na primer, necelobrojnu vrednost ili nenumerički tip. Stoga, dodatno možemo generisati sledeće ulazne klase ekvivalencije:

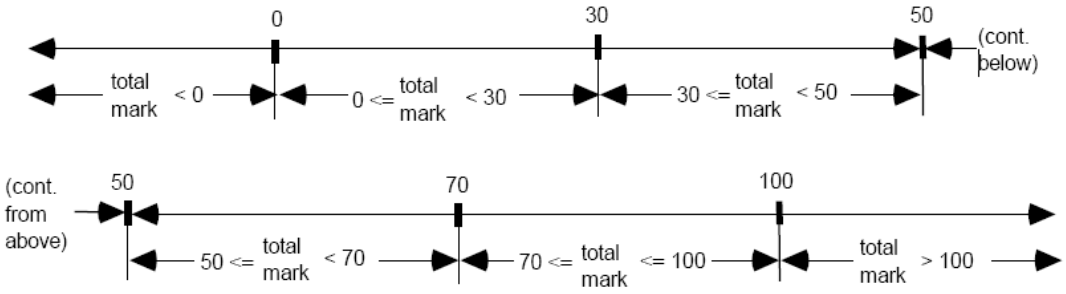
broj poena na ispitu = realan broj
broj poena na ispitu = alfabetski karakter
broj poena na domaćem = realan broj
broj poena na domaćem = alfabetski karakter

Nadalje, možemo identifikovati klase za izlaz. Validne klase ekvivalencije dobijamo razmatranjem svakog validnog izlaza komponente.

'A'	je posledica	70 <= ukupan broj poena <= 100
'B'	je posledica	50 <= ukupan broj poena < 70
'C'	je posledica	30 <= ukupan broj poena < 50
'D'	je posledica	0 <= ukupan broj poena < 30
'Fault Message'	je posledica	ukupan broj poena > 100
'Fault Message'	je posledica	ukupan broj poena < 0

gde je ukupan broj poena = broj poena na domaćem + broj poena na ispitu. Treba primetiti da je 'Fault Message' korektan izlaz jer je definisan kao jedan od mogućih izlaza u specifikaciji.

Klase ekvivalencije i granice za ukupni broj poena, prikazani su na narednoj slici:



Nekorektan izlaz komponente je bilo koji izlaz različit od 5 ranije navedenih. Teško je identifikovati nekorektan izlaz, ali ih svakako treba razmotriti jer ukoliko uspemo da izgenerišemo jedan, značilo bi da smo identifikovali grešku ili u komponenti ili u specifikaciji, a moguće i jedno i drugo.

Za naš primer identifikovana su tri nekorektna izlaza. Ovaj izbor klasa ekvivalencije je subjektivan i različiti testeri bi izabrali različite klase ekvivalencije, za koje oni osećaju da se mogu izgenerisati od strane generatora ocena.

izlaz = ‘E’
izlaz = ‘A+’
izlaz = ‘null’

Kada sumiramo, narednih 19 klasa ekvivalencije smo identifikovali (zapamtiti da ne bi svi testeri došli do identične liste klasa ekvivalencije):

0 <= broj poena na ispitu <= 75
0 <= broj poena na domaćem <= 25
broj poena na ispitu > 75
broj poena na ispitu < 0
broj poena na domaćem > 25
broj poena na domaćem < 0
broj poena na ispitu = realan broj
broj poena na ispitu = alfabetski karakter
broj poena na domaćem = realan broj
broj poena na domaćem = alfabetski karakter
70 <= ukupan broj poena <= 100
50 <= ukupan broj poena < 70
30 <= ukupan broj poena < 50
0 <= ukupan broj poena < 30
ukupan broj poena > 100
ukupan broj poena < 0
izlaz = ‘E’
izlaz = ‘A+’
izlaz = ‘null’

Nakon što smo identifikovali sve klase ekvivalencije, potrebno je da izgenerišemo odgovarajuće testove za svaku od njih. Dva različita pristupa su moguća kada se vrši generisanje testova. Jedan pristup jeste generisanje testa za svaku identifikovanu klasu posebno **(1 na 1).** Drugi pristup podrazumeva generisanje **minimalnog skupa testova** koji pokrivaju sve klase ekvivalencije.

U nastavku najpre će biti prikazan prvi pristup, kada generišemo jedan test za jednu klasu ekvivalencije, glavni razlog je što je ovaj pristup jednostavniji da se pokaže veza izmedju klasa ekvivalencije i testova. Za devetnaest izdvojenih particija izgenerisano je devetnaest test primera.

Testovi koji odgovaraju broju poena osvojenom na ispitu:

Broj test primera	1	2	3
Ulaz (poeni sa ispita - I)	44	-10	93
Ulaz (poeni sa domaćeg - D)	15	15	15
Ukupan broj poena (izračunava se automatski)	59	5	108
Particija koja je testirana	$0 \leq I \leq 75$	$I < 0$	$I > 75$
Očekivani izlaz	‘B’	‘FM’	‘FM’

Obratiti pažnju da je ulaz sa poenima sa domaćeg, postavljen na proizvoljnu, korektnu vrednost (na primer: 15).

Testovi koji odovaraju klasi ekvivalenciji koja opisuje ulaz, broj poena na domaćem:

Broj test primera	4	5	6
Ulaz (poeni sa ispita)	40	40	40
Ulaz (poeni sa domaćeg)	8	-15	47
Ukupan broj poena (izračunava se automatski)	48	25	87
Particija koja je testirana	$0 \leq D \leq 25$	$D < 0$	$D > 25$
Očekivani izlaz	‘C’	‘FM’	‘FM’

Primetiti da je ulaz broj poena osvojen na ispitu postavljena na proizvoljnu, korektnu vrednost (na primer: 40).

Testovi koji odgovaraju particijama za nekorektne ulazne vrednosti:

Broj test primera	7	8	9	10
Ulaz (poeni sa ispita)	48.7	q	40	40
Ulaz (poeni sa domaćeg)	15	15	12.76	g
Ukupan broj poena (izračunava se automatski)	63.7	nije moguće	52.76	nije moguće
Particija koja je testirana	I = realan broj	I = alfabetski karakter	D = realan broj	D = alfabetski karakter
Očekivani izlaz	‘FM’	‘FM’	‘FM’	‘FM’

Testovi koji odgovaraju klasama ekvivalencije izvedenih na osnovu korektnog izlaza:

Broj test primera	11	12	13
Ulaz (poeni sa ispita)	-10	12	32
Ulaz (poeni sa domaćeg)	-10	5	13
Ukupan broj poena (izračunava se automatski)	-20	17	45
Particija koja je testirana	Ukupno < 0	0 ≤ Ukup. < 30	30 ≤ Ukup. < 50
Očekivani izlaz	'FM'	'D'	'C'

Broj test primera	14	15	16
Ulaz (poeni sa ispita)	44	60	80
Ulaz (poeni sa domaćeg)	22	20	30
Ukupan broj poena (izračunava se automatski)	66	80	110
Particija koja je testirana	50 ≤ Ukup. < 70	70 ≤ Ukup. ≤ 100	Ukupno > 100
Očekivani izlaz	'B'	'A'	'FM'

Ulazne vrednosti za broj poena na ispitu i broj poena na domaćem dobijeni su na osnovu ukupnog broja poena.

Testovi koji odgovaraju klasama ekvivalencije izvedenih na osnovu nevalidnog izlaza:

Broj test primera	17	18	19
Ulaz (poeni sa ispita)	-10	100	null
Ulaz (poeni sa domaćeg)	0	10	null
Ukupan broj poena (izračunava se automatski)	-10	110	null + null
Particija koja je testirana	'E'	A+	null
Očekivani izlaz	'FM'	'FM'	'FM'

Treba zapaziti mesta gde je potrebno izgenerisati nevalidan ulaz (u našem primeru za slučajeve 2, 3, 5-11 i 16-19). U zavisnosti od implementacije, postoji mogućnost da nije moguće izgenerisati ove ulaze. Na primer, u jeziku Ada, ukoliko je promenljiva deklarisana kao pozitivna nije joj moguće dodeliti negativnu vrednost.

Ukoliko ponovo pogledamo prethodno izgenerisane testove, može se zapaziti da su pojedini testovi identični, na primer 1 i 14, dok je glavna razlika u klasama ekvivalencije za koju su predviđeni. Moguće je generisati testove koji bi pokrili veći broj particija istovremeno. U nastavku je data grupa testova koja pokriva sve ranije identifikovane particije.

Broj test primera	1	2	3	4
Ulaz (poeni sa ispita)	60	40	25	15
Ulaz (poeni sa domaćeg)	20	15	10	8
Ukupan broj poena (izračunava se automatski)	80	55	35	23
Particija za ispitne poene	$0 \leq I \leq 75$	$0 \leq I \leq 75$	$0 \leq I \leq 75$	$0 \leq I \leq 75$
Particija za poene sa domaćeg	$0 \leq D \leq 25$	$0 \leq D \leq 25$	$0 \leq D \leq 25$	$0 \leq D \leq 25$
Particija za ukupan br.poena	$70 \leq U \leq 100$	$50 \leq U \leq 70$	$30 \leq U \leq 50$	$0 \leq U \leq 30$
Očekivani izlaz	‘A’	‘B’	‘C’	‘D’

Broj test primera	5	6	7	8
Ulaz (poeni sa ispita)	-10	93	60.5	j
Ulaz (poeni sa domaćeg)	-15	35	20.23	k
Ukupan broj poena (izračunava se automatski)	-25	128	80.73	/
Particija za ispitne poene	$I < 0$	$I > 75$	I = realni broj	I = alfabetski karakter
Particija za poene sa domaćeg	$D < 0$	$D > 25$	D = realni broj	D = alfabetski karakter
Particija za ukupan br.poena	$U < 0$	$U > 100$	$70 \leq U \leq 100$	/
Očekivani izlaz	‘FM’	‘FM’	‘FM’	‘FM’

Broj test primera	9	10	11
Ulaz (poeni sa ispita)	-10	100	null
Ulaz (poeni sa domaćeg)	0	10	null
Ukupan broj poena (izračunava se automatski)	-10	110	null + null
Particija za ispitne poene	$I < 0$	$I > 75$	/
Particija za poene sa domaćeg	$0 \leq D \leq 25$	$0 \leq D \leq 225$	/
Particija za ukupan br.poena	$U < 0$	$U > 100$	/
Particija za izlaz	‘E’	‘A+’	null
Očekivani izlaz	‘FM’	‘FM’	‘FM’

Dva prikazana pristupa, **1-1** i **miminalni pristup**, predstavljaju dva ekstremuma u spektru mogućih pristupa za testiranje klasa ekvivalencije. Nedostatak 1-1 je u velikom broju testova koje je potrebno izgenerisati, ukoliko to predstavlja problem treba pristupiti minimalnom pristupu. Najčešće je pronalaženje klasa ekvivalencije dugotrajan proces u poređenju sa generisanjem testova, iz tog razloga bilo kakva ušteda u generisanju testova je relativno mala ukoliko je uporedimo sa ukupnom cenom. Nedostatak minimalnog pristupa je težina identifikovanja uzroka, jer se više klasa ekvivalencije testira istovremeno.

Zadatak 3: Prodavnica obuće

Neka je dat deo softverskog sistema koji služi za unos artikala u prodavnici obuće i spisak različitih veličina određene obuće. Specifikacija tekstualne datoteke koja se učitava u program navodi da artikli treba da budu alfabetski znaci od 2 do 15 karaktera. Svaka veličina može biti u opsegu od 26 do 55, i predstavlja se samo celim brojevima. Veličine se unose u rastućem poretku (prvo manje veličine). Maksimalno pet veličina se može uneti za svaki artikal. Format unosa je: naziv artikla, sledi zarez, a zatim sledi spisak veličina. Zarez se takođe koristi da razdvoji veličine. Blanko znaci treba da se zanemare na ulazu. Napisati sve klase ekvivalencije za ovaj deo softverskog sistema. Napisati skup test primera koji će pokriti ove klase ekvivalencije (za svaki test primer navesti ulazne podatke i očekivani izlaz).

Rešenje:

Klase ekvivalencije su:

1. Naziv artikla je slovo (ispravno)
2. Naziv artikla nije slovo (nije ispravno)
3. Dužina naziva artikla je manja od 2 slova (nije ispravno)
4. Dužina naziva artikla je između 2 i 15 karaktera (ispravno)
5. Dužina naziva artikla je veća od 15 karaktera (nije ispravno)
6. Veličina je manja od 26 (nije ispravno)
7. Veličina je u opsegu od 26 do 55 (ispravno)
8. Veličina je veća od 55 (nije ispravno)
9. Veličina je ceo broj (ispravno)
10. Veličina je realni broj (nije ispravno)
11. Veličina je upisana ciframa (ispravno)
12. Veličina sadrži karaktere koji nisu cifre (nije ispravno)
13. Veličine su unete u rastućem redosledu (ispravno)
14. Veličine su unete u opadajućem redosledu (nije ispravno)
15. Nisu unete veličine za neki artikal (nije ispravno)
16. Uneto je između jedne i pet veličina (ispravno)
17. Uneto je više od pet veličina (nije ispravno)
18. Naziv arikla je unet pre veličine (ispravno)
19. Naziv artikla nije unet pre veličine (nije ispravno)
20. Svaki podatak je razdvojen zarezom (ispravno)
21. Svaki podatak nije razdvojen zarezom (nije ispravno)
22. Ulazni podaci ne sadrže blanko znakove (???)
23. Ulazni podaci sadrže blanko znakove (????)

Testovi:

#	Test primeri	Očekivani izlaz	Pokrivene klase ekv.
1	xy,1	True	1,4,7,9,11,13,16,18,20,22
2	AbcDefghijklmno,1,2,3 ,4,48	True	1,4,7,9,11,13,16,18,20,23
3	a2x,1	False	2
4	A,1	False	3
5	Abcdefghijklmnop	False	5
6	Xy,0	False	6
7	XY,49	False	8
8	Xy,2.5	False	10
9	xy,2,1,3,4,5	False	14
10	Xy	False	15
11	XY,1,2,3,4,5,6	False	17
12	1,Xy,2,3,4,5	False	19
13	XY2,3,4,5,6	False	21
14	AB,2#7	False	12

Zadatak 4: Problem trougla

Neka je dat program koji učitava vrednosti iz 3 tekstualna polja (poljeA, poljeB i poljeC). U svako tekstualno polje korisnik treba da unese celobrojnu vrednost od 1 do 50, koja predstavlja stranicu trougla. Unos alfabetskih karaktera ili brojeva drugog tipa nije zabranjen. Kada se klikne na dugme za potvrdu, funkcija *proveri_trougao* ovog programa vraća kategoriju kojoj trougao pripada, odnosno jedan od sledećih izlaznih podataka:

- Nije trougao,
- Raznostrani,
- Jednakokraki,
- Jednakostranični.

Takođe, ukoliko je trougao moguće formirati, na izlazu se pojavljuje vrednost obima tog trougla, u suprotnom prikazuje se poruka o grešci. Metodom podele na klase ekvivalencije, definisati sve test primere u skladu sa opisanom specifikacijom programa.

Rešenje:

Klase ekvivalencije za ulazne podatke su:

1. Stranica A je celobrojna veličina iz traženog opsega (ispravno)
2. Stranica A je celobrojna veličina manja od 0 (nije ispravno)

3. Stranica A je celobrojna veličina veća od 50 (nije ispravno)
4. Stranica B je celobrojna veličina iz traženog opsega (ispravno)
5. Stranica B je celobrojna veličina manja od 0 (nije ispravno)
6. Stranica B je celobrojna veličina veća od 50 (nije ispravno)
7. Stranica C je celobrojna veličina iz traženog opsega (ispravno)
8. Stranica C je celobrojna veličina manja od 0 (nije ispravno)
9. Stranica C je celobrojna veličina veća od 50 (nije ispravno)
10. U poljeA unet je realan broj (nije ispravno)
11. U poljeA unet je alfabetski karakter (nije ispravno)
12. U poljeB unet je realan broj (nije ispravno)
13. U poljeB unet je alfabetski karakter (nije ispravno)
14. U poljeC unet je realan broj (nije ispravno)
15. U poljeC unet je alfabetski karakter (nije ispravno)

Klase ekvivalencije za izlazne podatke su:

1. Raznostrani trougao i izračunat obim takvog trougla. (ispravno)
2. Jednakokraki trougao i izračunat obim takvog trougla. (ispravno)
3. Jednakostranični trougao i izračunat obim takvog trougla. (ispravno)
4. Nije trougao. (ispravno)
5. Greška! Vrednost polja a nije u dozvoljenom opsegu vrednosti.
6. Greška! Vrednost polja b nije u dozvoljenom opsegu vrednosti.
7. Greška! Vrednost polja c nije u dozvoljenom opsegu vrednosti.
8. Greška! Nedozvoljen ulazni podatak.

Test primer	Ulaz			Izlaz	Pokriva klase ekvivalencije
	a	b	c		
TP 1	3	4	5	Raznostrani, O = 12	u1, u4, u7, i1
TP 2	2	2	3	Jednakokraki, O = 7	u1, u4, u7, i2
TP 3	5	5	5	Jednakostranični, O = 15	u1, u4, u7, i3
TP 4	4	1	2	Nije trougao.	u1, u4, u7, i4
TP 5	-1	5	5	Greška! a nije u opsegu.	u2, i5
TP 6	51	5	5	Greška! a nije u opsegu.	u3, i5
TP 7	5	-3	5	Greška! b nije u opsegu.	u5, i6
TP 8	5	55	5	Greška! b nije u opsegu.	u6, i6
TP 9	5	5	-5	Greška! c nije u opsegu.	u8, i7
TP 10	5	5	100	Greška! c nije u opsegu.	u9, i7
TP 11	3.5	4	5	Greška! Nedozvoljen ulaz.	u10, i8
TP 12	3	4.2	5	Greška! Nedozvoljen ulaz.	u12, i8
TP 13	3	4	5.123	Greška! Nedozvoljen ulaz.	u14, i8
TP 14	d	4	5	Greška! Nedozvoljen ulaz.	u11, i8
TP 15	3	j	5	Greška! Nedozvoljen ulaz.	u13, i8
TP 16	3	4	v	Greška! Nedozvoljen ulaz.	u15, i8

Napomena:

U specifikaciji zadatka nisu definisana pravila za formiranje trougla, koja takođe treba da ispoštujemo, jer ne možemo da pretpostavimo da je implementacija programa tačna. Tako bi recimo umesto test primera TP2 = (2, 2, 3) mogli da stavimo test primer (1, 4, 1), koji bi prema opisu programa na izlazu trebao da prikaže rezultat: Jednakokraki, O = 6.
Ipak, ovaj trougao je nemoguće konstruisati sa traženom trojkom stranica (1, 4, 1). Dakle, u ovom zadatku, kada se razvijaju testovi, ne treba zaboraviti i osnovno pravilo za formiranje trougla, koje glasi: Zbir dve, proizvoljno odabrane stranice, mora biti veći od treće stranice, u suprotnom to nije trougao. Odnosno, u implementaciji treba da se izvrši provera:

$(a, b, c) : a \geq b + c$
$(a, b, c) : b \geq a + c$
$(a, b, c) : c \geq a + b$

S obzirom da ne znamo implementaciju ovog programa i da se za sada baziramo na testiranju funkcionalnosti, ovde treba ispitati ispravnost programa testiranjem specijalnih slučajeva.

METOD GRANIČNIH SLUČAJEVA

Zadatak 1: Mtest

Mtest je program koji sređuje rezultate ispita za maksimalno 200 studenata. Ulaz je datoteka 'ocr' koja sadrži zapise dužine 80 znakova i po sledećoj organizaciji:

- prvi zapis je naslov koji se koristi za svaki izlazni tekst i ima formu:
  kolone 1-80 tekst naslova.

- sledeća grupa zapisa opisuje tačne odgovore i sastoji se iz jednog ili više zapisa po formi:
  kolone 1-3 ukupan broj pitanja (1-999) (samo na prvom zapisu grupe);
  kolone 10-59 tačni odgovori na pitanja 1-50, 51-100 itd;
  kolona 80 sadrži znak '2'.

- sledeća grupa zapisa se sastoji od jedne ili više podgrupa, od kojih svaka opisuje odgovore jednog studenta. Svaka podgrupa se sastoji od odgovarajućeg broja zapisa po formi:
  kolone 1-9 šifra studenta
  kolone 10-59 odgovori na pitanja 1-50, 51-100 itd;
  kolona 80 sadrzi znak '3'.

Izlaz je u formi 4 datoteke - izveštaja:
- izveštaj, sortiran po šifri studenta, sa konačnim ocenama studenata;
- isti izveštaj, samo sortiran po konačnim ocenama;
- statistički izveštaj po ocenama (srednje vrednosti, devijacije itd);
- statistički izveštaj po pitanjima (procenat tačnih odgovora, itd).

Testirati program metodom graničnih slučajeva

Rešenje:

Redom obrađujemo specifikaciju, uočavamo ulazne uslove, izlazne uslove i njihove granične slučajeve:

1 datoteka
(1) prazna datoteka (ostali slučajevi podrazumevaju punu)

2 slog naslova
(2) nedostaje
(3) 1 znak
(4) 80 znakova

3 slog tačnih odgovora i broj tačnih odgovora
(5) broj pitanja = 1
(6) broj pitanja = 50

(7) broj pitanja = 51
(8) broj pitanja = 999
(9) broj pitanja = 0
(10) broj pitanja nenumerik
(11) nedostaju slogovi tačnih odgovora
(12) 1 slog tačnih odgovora više
(13) 1 slog tačnih odgovora manje

4 slog studentskih odgovora
(14) 0 studenata
(15) 1 student
(16) 200 studenata
(17) 201 student
(18) neki student ima 1 slog odgovora, a ima 2 sloga tačnih odgovora
(19) taj student je prvi u datoteci
(20) taj student je poslednji u datoteci
(21) neki student ima 2 sloga odgovora, a ima 1 slog tačnih odgovora
(22) taj student je prvi u datoteci
(23) taj student je poslednji u datoteci

Redom obradjujemo izlazne uslove:
1 izveštaji 1 i 2 (izveštaji sortirani po šifri, odnosno oceni)
0 studenata (14)
1 student (15)
200 studenata (16)
(24) svi studenti imaju istu ocenu
(25) svi studenti imaju različite ocene
(26) neki, ali ne svi studenti imaju istu ocenu (zbog rangova)
(27) neki student ima ocenu 0
(28) neki student ima ocenu 100
(29) neki student ima najmanju moguću sifru (zbog sorta)
(30) neki student ima najveću moguću sifru (zbog sorta)
(31) broj studenata taman odgovara kapacitetu strane (zbog štampe)
(32) broj studenata za 1 veći od kapaciteta strane (zbog štampe)

2 izveštaj 3 (srednje vrednosti i std. devijacije ocena)
(33) svi studenti imaju 100% uspeh (maksimalni prosek)
(34) svi studenti imaju 0% uspeh (minimalni prosek)
(35) 1 student ima 0% a svi ostali 100% (maksimalna std. devijacija)
(36) svi studenti imaju isti uspeh (minimalna std. devijacija)

3 izveštaj 4 (procenti tacnih odgovora po pitanjima)
(37) svi studenti tačno odgovorili na prvo pitanje
(38) svi studenti netačno odgovorili na prvo pitanje
(39) svi studenti tačno odgovorili na poslednje pitanje
(40) svi studenti netačno odgovorili na poslednje pitanje

(41) broj pitanja taman odgovara kapacitetu strane
(42) broj pitanja za 1 veći od kapaciteta strane

Zadatak 2: Generator Ocena

Komponenta prihvata broj poena na ispitu (vrednost do 75) i broj poena osvojen na domaćem(c/w) (vrednost do 25), na onovu kojih generiše ocenu na predmetu u rangu od 'A' do 'D'. Ocena se računa na osnovu ukupnog broja poena koji se dobija kao suma ocene na ispit i domaćem:

veće ili jednako 70	- 'A'
veće ili jednako 50, ali manje od 70	- 'B'
veće ili jednako 30, ali manje od 50	- 'C'
manje od 30	- 'D'

Ukoliko je ukupan broj poena izvan očekivanog opsega, ili bilo koji od ulaza nije korektan, ('FM') se generiše kao izlaz. Svi ulazi su celobrojne vrednosti.

Najpre se odrede klase ekvivalencije a zatim se generišu odgovarajući test primeri.
Klase ekvivalencije moguće je definisati i za ulaze i za izlaze komponente! Takodje i validni i nevalidni ulazi i izlazi se razmatraju.

Rešenje:

Na početku, potrebno je identifikovati klase ekvivalencije, zatim se odrede granice ovih klasa ekvivalencije, na kraju kreiraju se testovi za odredjene granice. Klase ekvivalencije identifikuju se i za ulaz i za izlaz generatora, takodje razmatraju se i validni i nevalidni ulazi i izlazi.

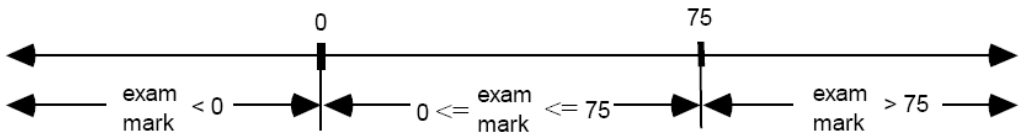
0 <= broj poena na ispitu <= 75
0 <= broj poena na domaćem <= 25

Najočiglednije nevalidne particije ulaza možemo opisati:

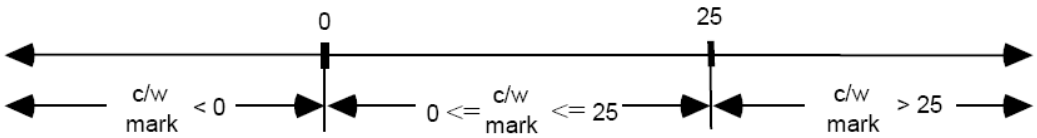
broj poena na ispitu > 75
broj poena na ispitu < 0
broj poena na domaćem > 25
broj poena na domaćem < 0

Identifikovane oblasti možemo prikazati grafički.

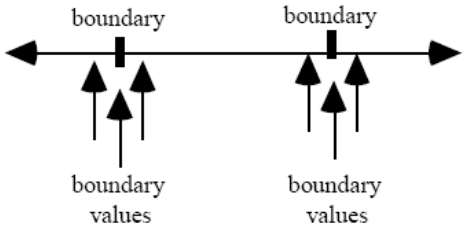
Što se tiče ulazne ocene sa ispita, grafički, oblasti može prikazati na sledeći način:



Slično, za ulaznu vrednost, broj poena na domaćem, imamo:



Za svaku granicu koriste se po tri vrednosti za testiranje, sama granična vrednosti i vrednosti sa obe strane granične vrednosti, najbliža moguća vrednost, kako je prikazano na slici:



S toga, šest vrednosti izvedenih na osnovu ulazne ocene sa ispita su:

Test Case	1	2	3	4	5	6
Input (exam mark)	-1	0	1	74	75	76
Input (c/w mark)	15	15	15	15	15	15
Boundary tested (exam mark)		0			75	
Exp. Output	'FM'	'D'	'D'	'A'	'A'	'FM'

Primetiti da je ulazna vrednost za broj poena osvojen na domaćem proizvoljno izabrana, iz skupa dozvoljenih vrednosti.

Na onovu ulaznih klasa ekvivalencije za broj poena osvojen na domaćem generisani su sledeći testovi:

Test Case	7	8	9	10	11	12
Input (exam mark)	40	40	40	40	40	40
Input (c/w mark)	-1	0	1	24	25	26
Boundary tested (c/w mark)		0			25	
Exp. Output	'FM'	'C'	'C'	'B'	'B'	'FM'

Primetiti da je ulazna vrednost za broj poena osvojen na ispitu, proizvoljno izabrana iz skupa dozvoljenih vrednosti.

Manje očigledne ulazne klase ekvivalencije mogu uključiti bilo koje ulaze koje nisu uključene u prethodne klase ekvivalencije. Na primer, necelobrojnu vrednost ili nenumerički tip. Stoga, dodatno možemo prepoznati sledeće ulazne klase ekvivalencije:

broj poena na ispitu = realan broj
broj poena na ispitu = alfabetski karakter
broj poena na domaćem = realan broj
broj poena na domaćem = alfabetski karakter

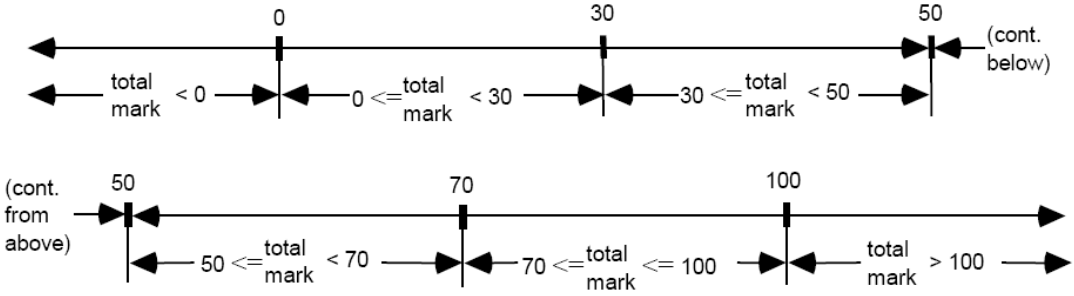
Iako su ovo validne klase ekvivalencije, za njih ne postoje granice koje se mogu identifikovati, te stoga uz njih nisu vezani testovi.

Nadalje, možemo identifikovati klase za izlaz. Validne klase ekvivalencije dobijamo razmatranjem svakog validnog izlaza komponente.

'A'	e posledica	70 <= ukupan broj poena <= 100
'B'	e posledica	50 <= ukupan broj poena < 70
'C'	e posledica	30 <= ukupan broj poena < 50
'D'	e posledica	0 <= ukupan broj poena < 30
'Fault Message'	e posledica	ukupan broj poena > 100
'Fault Message'	e posledica	ukupan broj poena < 0

gde je ukupan broj poena = broj poena na domaćem + broj poena na ispitu.

'Fault Message' se ovde razmatra kao specijalni izlaz. Klase ekvivalencije i granice za ukupni broj poena prikazani su na sledećoj slici:



Na osnovu prethodnog izgenerisani su testovi za validne izlaze:

Test Case	13	14	15	16	17	18	19	20	21
Input (exam mark)	-1	0	0	29	15	6	24	50	26
Input (c/w mark)	0	0	1	0	15	25	25	0	25
total mark (as calculated)	-1	0	1	29	30	31	49	50	51
Boundary tested (total mark)		0			30			50	
Exp. Output	'FM'	'D'	'D'	'D'	'C'	'C'	'C'	'B'	'B'

Test Case	22	23	24	25	26	27
Input (exam mark)	49	45	71	74	75	75
Input (c/w mark)	20	25	0	25	25	26
total mark (as calculated)	69	70	71	99	100	101
Boundary tested (total mark)		70			100	
Exp. Output	'B'	'A'	'A'	'A'	'A'	'FM'

Ulazne vrednosti za broj poena osvojen na ispitu i broj poena osvojen na domaćem izvedeni su iz ukupnog broja poena.

Nekorektan izlaz komponente je bilo koji izlaz različit od 5 ranije navedenih. Teško je identifikovati nekorektan izlaz, ali ih svakako treba razmotriti jer ukoliko uspemo da izgenerišemo jedan, značilo bi da smo identifikovali grešku ili u komponenti ili u specifikaciji, a moguće i jedno i drugo.

Za naš primer identifikovana su tri nekorektna izlaza (‘E’, ‘A+’, ‘null’). Medjutim nije moguće izvršiti uredjivanje ovih izlaza kako bismo definisali granične slučajeve, s toga nije moguće izgenerisati testove.

Do sada smo identifikovali nekoliko klasa koje su ograničene samo sa jedne strane. To su:

broj poena na ispitu	> 75
broj poena na ispitu	< 0
broj poena na domaćem	> 25
broj poena na ispitu	< 0
ukupni broj poena	> 100
ukupni broj poena	< 0

Činjenica je da su ove klase ekvivalencije na drugoj strani ograničene implementacionom maksimalnom i minimalnom vrednošću. Recimo za 16-obitni int to su 32767 i -32768.

Na osnovu poslednje konstatacije, prethodno prikazane klase mogu potpunije biti prikazane kao:

75 < broj poena na ispitu <= 32767
-32768 <= broj poena na ispitu < 0

25 < broj poena na domaćem <= 32767
-32768 <= broj poena na domaćem < 0
100 < ukupni broj poena <= 32767
-32768 <= ukupni broj poena < 0

Očigledno je, da je na ovaj način potrebno testirati i novouvedene granice. Ovo dovodi do sledećih test slučajeva:

Test Case	28	29	30	31	32	33
Input (exam mark)	32766	32767	32768	-32769	-32768	-32767
Input (c/w mark)	15	15	15	15	15	15
Boundary tested (exam mark)		32767			-32768	
Exp. Output	'FM'	'FM'	'FM'	'FM'	'FM'	'FM'

Test Case	34	35	36	37	38	39
Input (exam mark)	40	40	40	40	40	40
Input (c/w mark)	32766	32767	32768	-32769	-32768	-32767
Boundary tested (c/w mark)		32767			-32768	
Exp. Output	'FM'	'FM'	'FM'	'FM'	'FM'	'FM'

Test Case	40	41	42	43	44	45
Input (exam mark)	16383	32767	1	0	-16384	-32768
Input (c/w mark)	16383	0	32767	-32767	-16384	-1
total mark (as calculated)	32766	32767	32768	-32767	-32768	-32769
Boundary tested (total mark)		32767			-32768	
Exp. Output	'FM'	'FM'	'FM'	'FM'	'FM'	'FM'

Treba zapaziti mesta gde je potrebno izgenerisati nevalidan ulaz (u našem primeru za slučajeve 1, 6, 7, 12, 13, 27-45). U zavisnosti od implementacije, postoji mogućnost da nije moguće izgenerisati ove ulaze. Na primer, u jeziku Ada, ukoliko je promenljiva deklarisana kao pozitivna nije joj moguće dodeliti negativnu vrednost.

METOD UZROČNO-POSLEDIČNOG GRAFA
(eng. Cause-Effect Graph)

Zadatak 1: Potprogram IME

Dat je potprogram IME koji ispituje ispravnost imena datoteke u smislu broja znakova u identifikacionom i tipskom delu. Sintaksa imena je data sa:

<ime datoteke>.<tip>

gde je <ime datoteke> alfanumerički niz od 1 do 6 znakova, a <tip> se formira od 0 do 3 alfanumerička znaka. Ako je dužina tipa 0, podrazumeva se tip .DAT. Ulazni argumenti u potprogram su ime i tip datoteke, a izlaz je kod greške ER (ER=0 za ispravno ime, ER=1 za neispravno).
Odrediti test primere za testiranje potprograma IME metodom uzročno-posledičnog grafa.

Analiza problema

Koraci primene metoda:

1. Podela specifikacije na radne delove (da ograniči veličinu rezultujućeg grafa).
2. Identifikacija uzroka (ulazni uslov ili klasa ekvivalencije ulaznih uslova) i posledica (izlaz ili neka unutrašnja promena stanja sistema). Binarna vrednost: 0/1.
3. Analiziranjem značenja specifikacije konstruiše se uzročno-posledični graf. Čvorovima se predstavljaju uzroci, posledice i međučvorovi. Grane označavaju AND, OR i NOT zavisnosti među čvorovima, kao i ograničenja tipa I, E, O, R i M.

Ograničenja:

I – inkluzija, bar jedan od povezanih čvorova mora biti 1 (nedozvoljena kombinacija da su svi 0).

E – ekskluzija, najviše jedan od povezanih čvorova može biti 1 (nije dozvoljeno da više od jednog ima vrednost 1)

O – one&only one, tačno jedan od povezanih čvorova mora biti jedan

R – requires, ako strelica ide od čvora A ka čvoru B, to znači da bi A bilo 1 mora B biti 1 (nedozvoljena kombinacija B=0 i A=1, ostale su dozvoljene).

M – mask, ako strelica ide od čvora A ka čvoru B, to znači da B može imati vrednost 1 samo ukoliko A nema vrednost 1 (nedozvoljena kombinacija B=1 i A=1, ostale su dozvoljene).

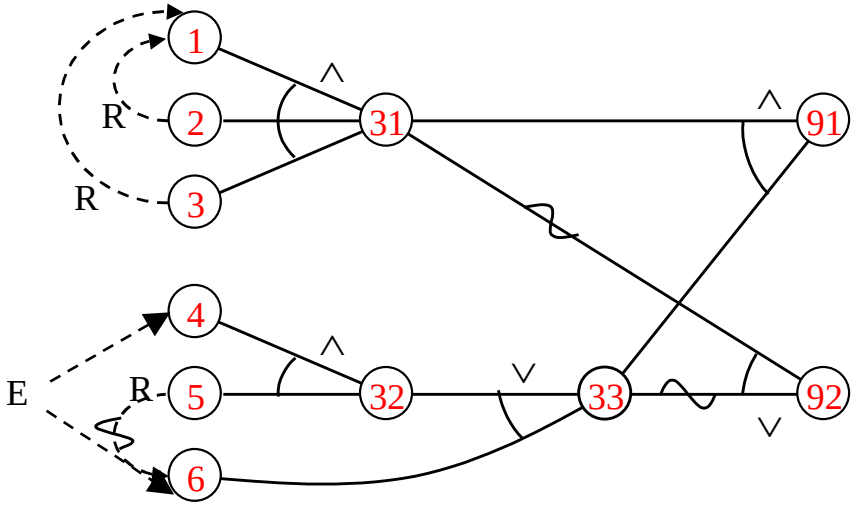
4. Konstrukcija tabele odlučivanja na osnovu uzročno posledičnog grafa.
5. Određivanje konkretnih test primera sa podacima na osnovu tabele odlučivanja

Rešenje:

Uzroci:
1. Dužina imena je veća od 0 znakova.
2. Ime datoteke sadrži samo alfanumeričke znake.
3. Dužina imena datoteke je manja ili jednaka 6 znakova.
4. Dužina tipa je od 1 do 3 znaka.
5. Tip sadrži samo alfanumeričke znake.
6. Dužina tipa je 0 znakova.

Posledice:
91. Ime datoteke je ispravno (ER=0).
92. Ime datoteke nije ispravno (ER=1).

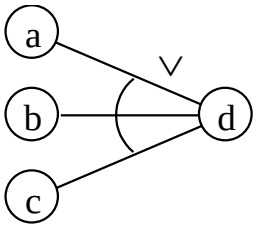
Međučvorovi:
31. Ispravna sintaksa za <ime datoteke>.
32. Ispravna sintaksa tipa za dužinu različitu od nule.
33. Ispravna sintaksa tipa u svim varijantama dužine tipa.



Pod ograničenjima uzroka ostalo 25 od 64 moguće komb. uzroka.

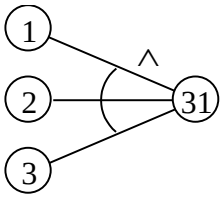
Transformacija grafa u tabelu odlučivanja:
1. Izabere se jedna posledica i podesi na 1.
2. Od izabrane posledice ide se unazad kroz graf i pronalaze kombinacije uzroka koji daju vrednost 1 posmatrane posledice.
3. Za svaku kombinaciju vrednosti uzroka formira se kolona u tabeli odlučivanja.
4. Za svaku od unetih kombinacija određuje se efekat na ostale posledice.

Meyers-ova pravila za redukovanje kombinacija uzroka:



d	a	b	c
∀0	∀0	∀0	∀0

d	a	b	c
∀1	∀0	∀0	∀1
	∀0	∀1	∀
	∀1	∀0	∀0



d	a	b	c
∀1	∀1	∀1	∀1

d	a	b	c
∀0	∃0	∃0	∃0
	∀0	∀0	∃1
	∀0	∃1	∀0
	∀0	∃1	∃1
	∃1	∀0	∀0
	∃1	∀0	∃1

Za dati graf imamo:

$\underline{91}_{\forall 1} \rightarrow \underline{31}_{\forall 1}\ \underline{33}_{\forall 1}$ $\qquad$ $\underline{31}_{\forall 1} \rightarrow \underline{1\ 2\ 3}_{1\ 1\ 1}$

$\underline{33}_{\forall 1} \rightarrow \begin{matrix}\underline{32} & \underline{6}\\ \forall 0 & 1\\ \forall 1 & 0\end{matrix}$ $\qquad$ $\underline{32}_{\forall 0} \rightarrow \underline{4}\ \underline{5}$

$0\ \ 0$

~~$0\ \ 1$~~ zbog ~R

~~$1\ \ 0$~~ zbog 4-E-6

$\underline{32}_{\forall 1} \rightarrow \underline{4\ 5}_{1\ 1}$

$\underline{92}_{\forall 1} \rightarrow \begin{matrix}\underline{31} & \underline{33}\\ \forall 0 & \forall 1\\ \forall 1 & \forall 0\end{matrix}$ $\qquad$ $\underline{31}_{\forall 0} \rightarrow \underline{1\ 2\ 3}$

$0\ 0\ 0$

~~$0\ 0\ 1$~~

~~$0\ 1\ 0$~~

~~$0\ 1\ 1$~~

$1\ 0\ 0$

$1\ 0\ 1$

$1\ 1\ 0$

$\underline{33}_{\forall 0} \rightarrow \underline{32}_{\forall 0}\ \underline{6}_{0}$

$\underline{32}_{\forall 0} \rightarrow \begin{matrix}\underline{4} & \underline{5}\\ 0 & 0\\ 0 & 1\\ 1 & 0\end{matrix}$ ništa se ne eliminiše

$\underline{31}_{\forall 1} \rightarrow \underline{1\ 2\ 3}_{1\ 1\ 1}$

čvor	TP1	TP2	TP3	TP4	TP5	TP6	TP7	TP8	TP9	TP10	TP11	TP12	TP13
1	1	1	0	0	1	1	1	1	1	1	1	1	1
2	1	1	0	0	0	0	0	0	1	1	1	1	1
3	1	1	0	0	0	0	1	1	0	0	1	1	1
4	0	1	0	1	0	1	0	1	0	1	0	0	1
5	0	1	0	1	0	1	0	1	0	1	0	1	0
6	1	0	1	0	1	0	1	0	1	0	0	0	0
91	1	1	0	0	0	0	0	0	0	0	0	0	0
92	0	0	1	1	1	1	1	1	1	1	1	1	1

Pravilima eliminisano 12 od mogućih 25 test primera.
Napomena: ako neki uzroci nisu definisani u nekoj koloni (ne utiču na vrednost posmatrane posledice) naknadno usvojiti neku vrednost tako da je ta kolona različita od ostalih.
Konkretni test primeri:

1. A123.
2. TEST.DAT
3. .
4. .D12
5. $$12345
6. .123456.DAT
7. $.
8. $.DAT
9. A1234567.
10. 1234567.DAT
11. A.$$$$
12. 12.DATOTEKA
13. 123456.$

Zadatak 2: Račun

Razmotrimo funkciju koja obrađuje podizanje novca. Ulazi funkcije su *količina novca* koja se podiže, *tip računa* i *trenutno stanje računa*, dok su izlazi *novo stanje računa* i *kod akcije*. Tip računa može biti poštanski (p) ili klasični (c). Kod akcije može biti ‘D&L’, ‘D’, ‘S&L’ ili ‘L’, tj. 'obradi podizanje novca i pošalji pismo’, ‘samo obradi podizanje novca’, 'blokiraj račun i pošalji pismo’ i 'samo pošalji pismo’, respektivno. Specifikacija funkcije data je u nastavku:
Ukoliko ima dovoljno novčanih sredstava na računu ili bi novo stanje bilo u granicama dozvoljenog minusa, podizanje novca se obradjuje. Ukoliko bi podizanje novca dovelo do prekoračenja dozvoljenog minusa, podizanje novca nije moguće, dodatno ukoliko je u pitanju poštanski račun vrši se privremeno blokiranje istog. Pismo se šalje za sve obavljene transakcije u slučaju poštanoskog računa, kao i za ne-poštanski račun ukoliko nema dovoljno novčanih sredstava (tj. račun više nije u plusu).

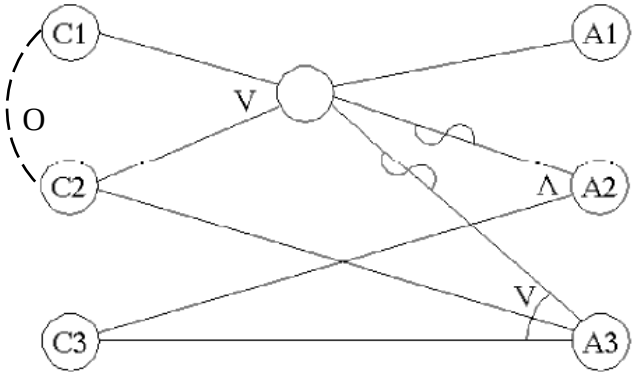
Uzroci:

1. Novo stanje je u plusu
2. Novo stanje je u dozvoljenom minusu
3. Račun je poštanski

Posledice:

1. Obrada podizanja novca
2. Privremeno blokiranje računa
3. Slanje pisma

Uzročno posledični graf prikazuje veze izmedju uzroka i posledica, na sličan način kao što to rade dizajneri hardverskih logičkih kola. Prethodno data specifikacija modelovana je grafom prikazanim na narednoj slici:



Dobijeni graf je potrebno predstaviti tabelom odlučivanja. Svaka kolona tabele odlučivanja odgovara funkcionalnom test primeru. Tabela se sastoji iz dva dela. U prvom delu svako pravilo je uredjeno prema uslovima. 'T' implicira da uslov mora biti TRUE kako bi se pravilo postiglo, dok 'F' implicira da uslov mora biti FALSE kako bi se pravilo primenilo. U drugom delu, svako pravilo je uredjeno na osnovu posledica (akcija). 'T' implicira da će akcija biti izvršena, dok 'F' implicira da akcija neće biti izvršena. '*' govori da je kombinacija uzroka nemoguća te stoga akcije nisu definisane za pravilo.

Za primer iz zadatka imamo sledeću tabelu odlučivanja:

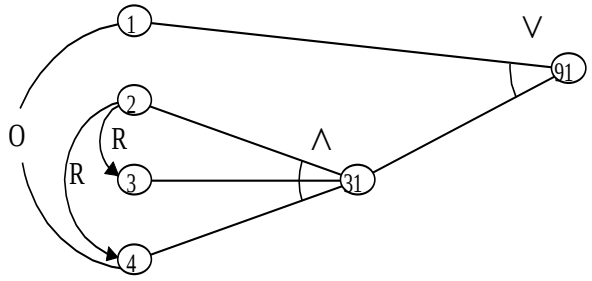
Test primeri:	1	2	3	4	5	6	7	8
C1: Novo stanje u plusu	F	F	F	F	T	T	T	T
C2: Novo stanje u dozvoljenom minusu	F	F	T	T	F	F	T	T
C3: Račun poštanski	F	T	F	T	F	T	F	T
A1: Obrada podizanja novca	F	F	T	T	T	T	*	*
A2: Privremeno blokiranje računa	F	T	F	F	F	F	*	*
A3: Slanje pisma	T	T	T	T	F	T	*	*

Testovi prikazani u nastavku se zahtevaju ukoliko se želi postizanje 100% pokrivenosti uzroka-posledica (Kako su pravila 7 i 8 nemoguća za njih nisu generisani test primeri).

Test primeri	UZROCI				POSLEDICE	
	tip računa	prekoračen limit	tekuće stanje	iznos zaduženja	novo stanje	kod akcije
1.	c	€ 100	-€ 70	€ 50	-€ 70	L
2.	p	€ 1 500	€ 420	€ 2 000	€ 420	S&L
3.	c	€ 250	€ 650	€ 800	-€ 150	D&L
4.	p	€ 750	-€ 500	€ 200	-€ 700	D&L
5.	c	€ 1 000	€ 2 100	€ 1 200	€ 900	D
6.	p	€ 500	€ 250	€ 150	€ 100	D&L

Zadatak 3: Tabela odlučivanja

Odrediti tabelu odlučivanja za dati uzročno-posledični graf.



Rešenje:

1	2	3	4	Ograničenje
~~0~~	~~0~~	~~0~~	~~0~~	O
0	0	0	1	
~~0~~	~~0~~	~~1~~	~~0~~	O

0	0	1	1	
~~0~~	~~1~~	~~0~~	~~0~~	O
~~0~~	~~1~~	~~0~~	~~1~~	R
~~0~~	~~1~~	~~1~~	~~0~~	O
0	1	1	1	
1	0	0	0	
~~1~~	~~0~~	~~0~~	~~1~~	O
1	0	1	0	
~~1~~	~~0~~	~~1~~	~~1~~	O
~~1~~	~~1~~	~~0~~	~~0~~	R
~~1~~	~~1~~	~~0~~	~~1~~	O
~~1~~	~~1~~	~~1~~	~~0~~	R
~~1~~	~~1~~	~~1~~	~~1~~	O

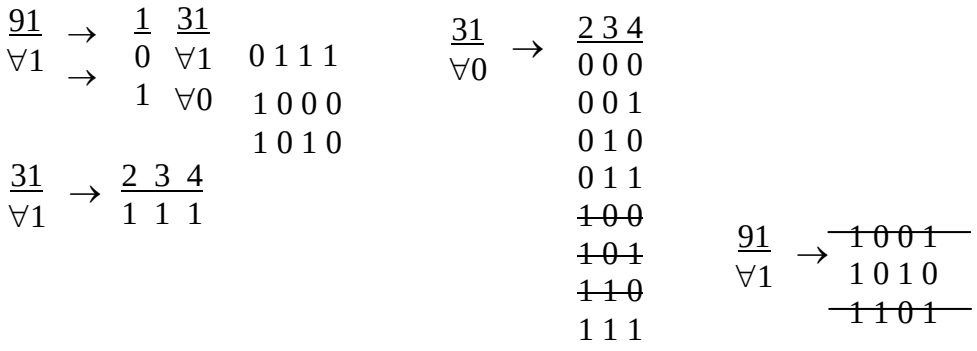
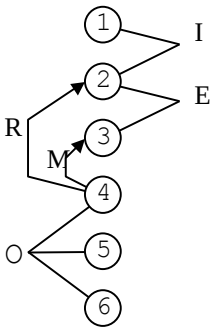


Tabela odlučivanja:

Čvor	Dozvoljene kombinacije uzroka		
uzrok 1	**0**	**1**	**1**
uzrok 2	**1**	**0**	**0**
uzrok 3	**1**	**0**	**1**
uzrok 4	**1**	**0**	**0**
posledica 91	**1**	**1**	**1**

Zadatak 4: Kombinacije uzroka

Na slici je prikazan deo uzročno-posledičnog grafa SAMO sa čvorovima uzroka i međusobnim ograničenjima. Navesti sve dozvoljene kombinacije uzroka.



Rešenje:

Dozvoljene su sledeće kombinacije uzroka:

Indeksi\	1	2	3	4	5	6
17	F	T	F	F	F	T
18	F	T	F	F	T	F
20	F	T	F	T	F	F
33	T	F	F	F	F	T
34	T	F	F	F	T	F
41	T	F	T	F	F	T
42	T	F	T	F	T	F
49	T	T	F	F	F	T
50	T	T	F	F	T	F
52	T	T	F	T	F	F

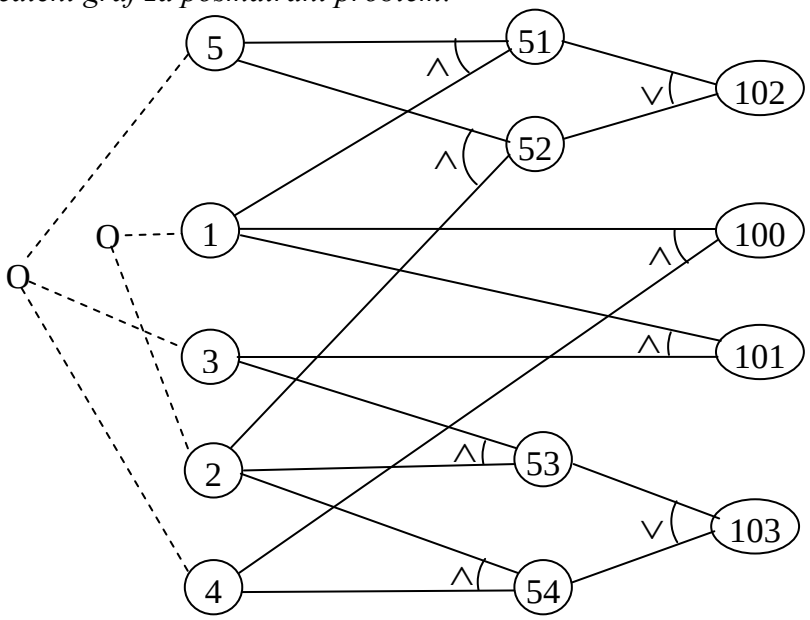
Zadatak 5 - Tehnika senzitizacije putanja
(eng. Basic Path-sensitization Technique)
*po urađenom primeru sa predavanja

Pravila izračunavanja premije osiguranja kola:
R00103 Za žene mlađe od 65 godina, premija je $500.
R00101 Za muškarce mlađe od 25 godina, premija je $3000.
R00100 Za muškarce između 25 i 64 godine, premija je $1000.
R00102 Za sve starosti 65 godina i više, premija je $1500.
Primeniti metod uzročno-posledičnog grafa na gornji skup zahteva za softver koji računa premije osiguranja. Koristiti BPST metod određivanja tabele odlučivanja.

Rešenje
Identifikujemo uzroke i posledice:

Uzroci	Posledice
1. Pol je muški	100. Premija je $1000
2. Pol je ženski	101. Premija je $3000
3. Godine su < 25	102. Premija je $1500
4. Godine su ≥ 25 i < 65	103. Premija je $500
5. Godine su ≥ 65	

Postoje ograničenja ONE AND ONLY ONE između uzroka 1 i 2, kao i između uzroka 3, 4 i 5.
Uzročno-posledični graf za posmatrani problem:



BPST tehnika
Uzročni skup (Cause Set) neke posledice je skup svih uzroka koji utiču na posmatranu posledicu.
Senzitizacija posledice na uzrok, znači da se svi ostali uzroci iz njenog uzročnog skupa postavljaju na takve vrednosti da posmatrana posledica zavisi direktno od uzroka ili njegove negacije.

BPST algoritam računanja tabele odlučivanja

```
For each effect, ej in a cause-effect graph CEG do
  For each cause cj in the Cause Set C of effect ej do
    ->Sensitize effect ej to cause cj
    ->For each possible combinations create two cause combinations one
       with cj set to 1 and another one with cj set to 0
    ->Create a column in the decision table for each cause combination
    ->Use the values set to the causes in C and the constraints present in the
CEG
   to determine the values of the other causes in the cause effect graph
    ->Use the values set to the causes in C in order to determine the values
  of the other effects in CEG for these two combinations
    ->Remove any redundant column
  End for
  ->If possible, create two additional cause combinations where in one most of
the causes    in C are absent and in the other most of them are present.
  ->Create a column in the decision table for each cause combination
  ->Use the values set to the causes in C and the constraints present in the
CEG to       determine the values of the other causes in the CEG
  ->Use the values set to the causes in C in order to determine the values of
he other     effects in CEG for these two combinations
  ->Remove any redundant column
End for
->Derive test scenarios from the decision table obtained
```

CauseSet(100) = {1,4}
Senzitizacija 100 na 1: mora 4=1, daje kombinacije 1 i 2, za 1=0 i 1=1 respektivno. Zbog ograničenja među uzrocima, ostali uzroci imaju fiksne vrednosti.
Senzitizacija 100 na 4: mora 1=1, što daje kombinacije 3 i 2, za 4=0 i 4=1 respektivno. Uzrok 2 je fiksiran na not(1), ali uzroci 3 i 4 nisu definisani. Na kraju, kada se konstrukcija tabele završi, treba im dodeliti slučajne vrednosti.
Treba još dodati kombinaciju broj 4 za maksimum nula za CauseSet(100). Takođe, kombinaciju za maksimum jedinica za CauseSet(100), dakle uzrok 1 i 4 su na 1, što već imamo u kombinaciji 2.

	1	2	3	4															
1	**0**	**1**	1	0															
2	1	0	0	1															
3	0	0	X	X															
4	1	1	**0**	0															
5	0	0	X	X															
100	**0**	**1**	**0**	0															
101	0	0	X	0															
102	0	0	X	X															
103	1	0	0	X															

Napomena: Plavim su označeni uzrok i posledica koji su senzitizovani u posmatranoj kombinaciji.

CauseSet(101)={1,3}
Senzitizacija 101 na 1 i 3 daje sledeće kombinacije (kolona 8 je za maksimum nula u CauseSet-u, a kolona 6 za maksimum jedinica):

	5	6	7	8															
1	**0**	**1**	1	0															
2	1	0	0	1															
3	1	1	**0**	0															
4	0	0	X	X															
5	0	0	X	X															
100	0	0	X	0															
101	**0**	**1**	**0**	0															
102	0	0	X	X															
103	1	0	0	X															

CauseSet(102)={1,2,5}
Senzitizacija 102 na 1: Uzrok 5=1, a međučvor 52=0, što znači da je 2=0. Za 1=0 dobija se kombinacija 9. Senzitizacija 102 na 1 kada je 1=0, otpada, jer bi trebalo da je 2=0 što ne može istovremeno kada je 1=0 (usled 1-O-2 ograničenja). Iz istog razloga otpada senzitizacija 102 na 2, za 2=0.
Senzitizacija 102 na 5: Imamo dve putanje od 5 ka posledici 102. Za senzitizaciju treba da izaberemo jednu. Ako uzmemo onu preko međučvora 51, onda uzrok 1=1, a uzrok 2=0.
Za 5=0 dobija se kombinacija 12, a za 5=1 već imamo kombinaciju 9.
Kombinacija za maksimum nula u CauseSet-u je 12 (jer 1-O-2), a kombinacija za maksimum jedinica u CauseSetu poklapa se sa 9 ili 10.

	9	10	11	12															
1	**1**	0	1	1															
2	0	**1**	0	0															
3	0	0	0	X															
4	0	0	0	X															
5	1	1	**1**	0															
100	0	0	0	X															
101	0	0	0	X															
102	**1**	**1**	**1**	0															
103	0	0	0	0															

CauseSet(103)={2,3,4}
Kod senzitizacije 103 na 2 biramo jednu od putanja od 2 ka 103, neka je to gornja.
Senzitizacija 103 na 2, 3 i 4 daje sledeće kombinacije:

	13	14	15	16	18														
1	0	1	0	0	1														
2	**1**	**0**	1	1	0														
3	1	1	**0**	0	0														
4	0	0	0	**1**	0														
5	0	0	1	0	1														

100	0	0	0	0	0														
101	0	1	0	0	0														
102	0	0	1	0	1														
103	**1**	**0**	**0**	**1**	0														

Konačno, treba izabrati vrednosti za X i potom izbaciti duplicirane kolone. Bez obzira koje vrednosti usvojili, u ovom konkretnom primeru ostaće ukupno sledećih 6 kombinacija, uzimajući u obzir O ograničenja među uzrocima (izvršeno je prenumerisanje kolona):

	1	2	3	4	5	6
1	0	1	0	0	1	**1**
2	**1**	**0**	1	1	0	0
3	1	1	**0**	0	0	0
4	0	0	0	**1**	0	1
5	0	0	1	0	1	0
100	0	0	0	0	0	**1**
101	0	1	0	0	0	0
102	0	0	1	0	1	0
103	**1**	**0**	**0**	**1**	0	0

KOMBINATORNO TESTIRANJE

(eng. Combinatorial Software Testing Techniques)

All-Pairs (Pairwise) metod testiranja

Teorija (Osnove): Dekartov proizvod

Dobra startna tačka za diskusiju o all-pairs je Dekartov proizvod. Dekartov proizvod je scenario u kome je svaka jedinica iz grupe povezana sa svakom jedinicom iz druge grupe, tako da su sve kombinacije jedinica kreirane uključujući sve grupe.

Na primer, zamislimo sledeću jednostavnu veb aplikaciju sa jednim ekranom, na kojoj se nalaze dve padajuće liste i dugme 'OK'. Lista 1 sadrži tri vrednosti: 0, 1 i 2. Lista 2 sadrži dve vrednosti: 10 i 20. Korisnik selektuje jednu od vrednosti za svaku listu, nakon klika na dugme 'OK' proizvod dve unete vrednosti se prikazuje na ekranu.

Promenljive i njihove vrednosti izgledaju kao na narednoj slici:

Lista 1	Lista 2
0	10
1	20
2	

Kako biste testirali ovaj program? Koliko različitih ulaznih kombinacija postoji? Koliko vremena bi bilo potrebno za testiranje svih mogućih kombinacija.

Postoji 3 x 2 = 6 mogućih kombinacija. Šest rezultujućih kombinacija je rezultat Dekatovog prozvoda dva ulaza i izgledao bi ovako:

Lista 1	Lista 2
0	10
0	20
1	10
1	20
2	10
2	20

Svaka vrednost, svake promenljive je upotrebljena tako da su kreirane sve kombinacije. Ovaj test možemo izvršiti i ručno za manje od jednog minuta. Sada ćemo da zakomplikujemo naš primer.

b) Verzija 2.0 našeg programa ima nove karakteristike:

Lista 1 sada sadrži int vrednosti od 0 do 9 i Lista 2 je promenjena u tekstualno polje, dopuštajući unos int vrednosti izmedju 1 i 99. Dodali smo i dva polja za setovanje (*checkbox*). Kada je čekirano prvo polje, dobijeni proizvod množi sa -1 (vrši negaciju). Kada je čekirano drugo polje, vrši se množenje dobijenog proizvoda sa samim sobom (generiše se kvadrat dobijenog proizvoda). Promenljive i njihove vrednosti sada izgledaju ovako:

Lista	Textbox	Checkbox1 (-x)	Checkbox2 (x*x)
0	1	On	On
1	2	Off	Off
2	3		
3	4		
4	...		
5	96		
6	97		
7	98		
8	99		
9			

Sada možemo postaviti isto pitanje: Kako bi ste testirali ovaj program? Koliko različitih kombinacija ulaza postoji? Koliko vremena je potrebno za testiranje svih kombinacija?

U ovom slučaju imamo 10 x 99 x 2 x 2 = 3 960 mogućih korektnih ulaznih vrednosti. Takođe postoji i određeni broj nekorektnih ulaza (obratite pažnju da je tekstualno polje uvelo novi koncept - mogućnost nevalidnog ulaza, što je diskutovano kasnije).

Kako rešiti probleme?

Prvo je potrebno da identifikujemo parametre (promenljive) i vrednosti koje svaka promenljiva može uzeti. Potrebno je organizovati ove informacije u tabele. Sledeći korak je da pojednostavimo vrednosti. Neophodno je grupisati vrednosti u nekoliko kategorija (grupa) i uvesti pojam graničnih vrednosti. Na primer, za slučaj tekstualnog polja, umesto listanja svih validnih vrednosti od 1 do 99, koristiti reprezentativnu vrednost (više o ovim informacijama pogledati materijale vezane za klase ekvivalencije i granične slučajeve). U našem primeru, korisnik ima više slobode za unos nevalidne vrednosti u tekstualno polje nego što je to slučaj sa padajućom listom. Primetiti da je vrednosti moguće grupisati čak i ukoliko nisu diskretne. Na primer, vrednost kategorisane za proizvoljni tekst mogu biti: svi alfabetski karakteri, svi numerički karakteri, velika slova, mala slova, reči pod navodnicima, itd.

U našem primeru, kako bi ga održali jednostavnim, grupišimo 99 korektnih vrednosti, plus beskonačan broj nekorektnih vrednosti u tri grupe: bilo koja validna vrednost, bilo koja nevalidna vrednost, bilo koji alfabetski karakter (koji bi bilo nevalidan). Takođe, grupišimo 10 vrednosti u padajućoj listi na dve: 0 i druga vrednost. Sada naše promenljive i vrednosti izgledaju ovako:

Lista	Textbox	Checkbox1 (-x)	Checkbox2 (x*x)
0	Validni int	On	On
Bilo koja vrednost	Nevalidni int	Off	Off
	Alfabetski karakter		

Sledeće što ćemo uraditi je da stavimo vrednosti u prvu vrstu tabele. Uredimo promenljive tako da je promenjiva sa najviše mogućih vrednosti prva i da je promenljiva sa najmanje vrednosti poslednja. U ovom primeru najpre stavljamo tekstualno polje, jer poseduje tri moguće vrednosti. Sve ostale promenljive mogu uzeti po dve vrednosti, pa redosled nije bitan.

Textbox (3)	Lista (2)	Negativni proizvod (2)	Kvadrat proizvoda (2)

Sada možemo početi sa popunjavanjem tabele. Svaka vrsta tabele predstavljaće jedinstveni test primer. Popunjavaćemo kolonu po kolonu. Pogledajmo koliko

vrednosti imamo za kolonu 2. Ovde vidimo da kolona vezana za listu ima 2 vrednosti. Ovim smo odredili koliko puta je potrebno da umetnemo vrednosti za prvu kolonu, tekst polje. Da počnemo:

Textbox (3)	Lista (2)	Negativni proizvod (2)	Kvadrat proizvoda (2)
Validni ceo broj			
Validni ceo broj			
Nevalidni ceo broj			
Nevalidni ceo broj			
Alfabetski karakter			
Alfabetski karakter			

Umetnuli smo šest vrsta. Tri vrednosti za tekst polje, svaka se pojavljuje po dva puta. Primetiti da smo preskočili po jednu vrstu između svakog skupa vrednosti. Ovo je važno - uskoro sledi objašnjenje. Sada možemo popuniti kolonu 2. Za svaki skup vrednosti u koloni 1, dodaćemo obe vrednosti kolone 2:

Textbox (3)	Lista (2)	Negativni proizvod (2)	Kvadrat proizvoda (2)
Validni ceo broj	0		
Validni ceo broj	Bilo koja druga		
Nevalidni ceo broj	0		
Nevalidni ceo broj	Bilo koja druga		
Alfabetski karakter	0		
Alfabetski karakter	Bilo koja druga		

Pripremili smo vrednosti za naše prve dve promenljive. Možemo kratko proveriti... Korektna vrednost i 0, korektna vrednost i „Bilo koja druga”, nekorektna vrednost i 0, nekorektna vrednost i „Bilo koja druga”, alfabetski karakter i 0, alfabetski karakter i „Bilo koja druga”.

Razmotrimo treću promenljivu. Kolona tri jeste polje za čekiranje koje određuje da li želimo da izvršimo množenje dobijenog rezultata sa -1. Postoje dve vrednosti, uključeno i isključeno. Hajde da stavimo na uključeno i isključeno u koloni 3 i da vidimo šta se dešava.

Textbox (3)	Lista (2)	Negativni proizvod (2)	Kvadrat proizvoda (2)
Validni ceo broj	0	*On*	
Validni ceo broj	Bilo koja druga	*Off*	
Nevalidni ceo broj	0	*On*	
Nevalidni ceo broj	Bilo koja druga	*Off*	
Alfabetski karakter	0	*On*	
Alfabetski karakter	Bilo koja druga	*Off*	

Hajde da proverimo da li imamo sve moguće parove izmedju kolona 3 i 2. Imamo 0 i ON, ali nemamo 0 i OFF. Takođe imamo „Bilo koja druga" i OFF, ali nema „Bilo koja druga" i ON. Hajde da zamenimo vrednosti u drugom skupu treće kolone:

Textbox (3)	Lista (2)	Negativni proizvod (2)	Kvadrat proizvoda (2)
Validni ceo broj	0	*On*	
Validni ceo broj	Bilo koja druga	*Off*	
Nevalidni ceo broj	0	*Off*	
Nevalidni ceo broj	Bilo koja druga	*On*	
Alfabetski karakter	0	*On*	
Alfabetski karakter	Bilo koja druga	*Off*	

Znatno bolje. Sada imamo 0/ON, 0/OFF, „Bilo koja druga"/ON i „Bilo koja druga"/OFF. Primetimo da je poslednji skup ON i OFF proizvoljan - već imamo parove koje smo želeli i nije nam važno da li je redosled ON/OFF ili OFF/ON. Nastavimo sa četvrtom kolonom. Ovo je polje za selektovanje koje govori da li će se izvršiti kvadriranje dobijenog proizvoda. Takođe postoje dve vrednosti, selektovano i nije selektovano. Potrebno je da unesemo vrednosti za ovo polje na način da dobijemo kombinaciju svih parova vrednosti. Da probamo na sledeći način:

Textbox (3)	Lista (2)	Negativni proizvod (2)	Kvadrat proizvoda (2)
Validni ceo broj	0	*On*	Čekiran
Validni ceo broj	Bilo koja druga	*Off*	Nije čekiran
Nevalidni ceo broj	0	*Off*	Čekiran
Nevalidni ceo broj	Bilo koja druga	*On*	Nije čekiran
Alfabetski karakter	0	*On*	Nije čekiran
Alfabetski karakter	Bilo koja druga	*Off*	Čekiran

Još jednom proverimo naše parove. Imamo 0/Čekiran i 0/Nije čekiran. Takođe imamo „Bilo koja druga"/Čekiran i „Bilo koja druga"/Nije čekiran. Sada pogledajmo kolone 3 i 4. Imamo ON/Čekiran i ON/Nije čekiran i OFF/Čekiran i OFF/Nije čekiran. Uspeli smo da uklopimo sve parove naših vrednosti u šest slučajeva. Da smo testirali sve kombinacije vrednosti imali bi smo 3 x 2 x 2 x 2 = 24 kombinacije. Ukoliko razmotrimo da smo izvršili redukovanje sa 99 na 3 u slučaju tekstualnog polja, kao i sa 10 na 2 vrednosti u slučaju padajuće liste, dolazimo do toga da je ušteda čak i veća.

c) Setite se da smo pomenuli važnost linija koje smo ostavili praznim ranije. Recimo da verzija 3.0 našeg programa doda još dva polja za selektovanje. Polje za selektovanje 3 će dati faktorijel za dobijeni proizvod, a polje za selektovanje 4 će konvertovati izlaz u heksadecimalnu notaciju. Na osnovu izmena potrebno je da dodamo još dve kolone u naše tabele i unesemo njihove vrednosti. Najpre razmotrimo polje za selektovanje 3 u petoj koloni:

Textbox (3)	**Lista (2)**	**Negativni proizvod (2)**	**Kvadrat proizvoda (2)**	**Factorial (2)**	**Hex (2)**
Validni ceo broj	0	*On*	Čekiran	Da	
Validni ceo broj	Bilo koja druga	*Off*	Nije čekiran	Ne	
Nevalidni ceo broj	0	*Off*	Čekiran	Ne	
Nevalidni ceo broj	Bilo koja druga	*On*	Nije čekiran	Da	
Alfabetski karakter	0	*On*	Nije čekiran	Ne	
Alfabetski karakter	Bilo koja druga	*Off*	Čekiran	Da	

Utvrdimo da li svaka kolona ima bar jedan par sa našom novo dodatom, petom kolonom. Izgleda da je sve OK. Primetiti da poslednji skup vrednosti u trećoj koloni nije više proizvoljan, kao što je bio ranije.

Postavimo i vrednosti za poslednju kolonu:

Textbox (3)	**Lista (2)**	**Negativni proizvod (2)**	**Kvadrat proizvoda (2)**	**Factorial (2)**	**Hex (2)**
Validni ceo broj	0	*On*	Čekiran	Da	Dec
Validni ceo broj	Bilo koja druga	*Off*	Nije čekiran	Ne	Hex

Nevalidni ceo broj	0	*Off*	Čekiran	Ne	Hex
Nevalidni ceo broj	Bilo koja druga	*On*	Nije čekiran	Da	Dec
Alfabetski karakter	0	*On*	Nije čekiran	Ne	Dec
Alfabetski karakter	Bilo koja druga	*Off*	Čekiran	Da	Hex

Pogledajmo rezultat:

Kolona 2 je OK. Imamo 0/Dec, 0/Hex, „Bilo koja druga"/Dec i „Bilo koja druga"/Hex. Kolona 3 je problematična: imamo ON/Dec i OFF/Hex, ali nedostaju ON/Hex i OFF/Dec parovi. Kolona 4 je OK: Čekiran/Dec, Čekiran/Hex, Nije čekiran/Dec, Nije čekiran/Hex. Kolona 5 je OK: imamo Da/Dec, Da/Hex, Ne/Dec i Ne/Hex.

Ovaj put nije moguće dobiti nedostajući par izmenom redosleda nekih vrednosti. Umesto toga, jednostavno umećemo dva nova testa koja sadrže nedostajući par. Iz tog razloga smo imali prazne vrste.

Textbox (3)	**Lista (2)**	**Negativni proizvod (2)**	**Kvadrat proizvoda (2)**	**Factorial (2)**	**Hex (2)**
Validni ceo broj	0	*On*	Čekiran	Da	Dec
Validni ceo broj	Bilo koja druga	*Off*	Nije čekiran	Ne	Hex
		On			**Hex**
Nevalidni ceo broj	0	*Off*	Čekiran	Ne	Hex
Nevalidni ceo broj	Bilo koja druga	*On*	Nije čekiran	Da	Dec
		Off			**Dec**
Alfabetski karakter	0	*On*	Nije čekiran	Ne	Dec
Alfabetski karakter	Bilo koja druga	*Off*	Čekiran	Da	Hex

Ostale promenljive su potpuno proizvoljne. One moraju biti popunjene sa nekim vrednostima, ali nije bitno koje vrednosti su u pitanju. Tako dolazimo do All-pairs sa osam slučajeva, umesto svih kombinacija 96!

Textbox (3)	Lista (2)	Negativni proizvod (2)	Kvadrat proizvoda (2)	Factorial (2)	Hex (2)
Validni ceo broj	0	*On*	Čekiran	Da	Dec
Validni ceo broj	Bilo koja druga	*Off*	Nije čekiran	Ne	Hex
Validni ceo broj	Bilo koja druga	*On*	Nije čekiran	Ne	Hex
Nevalidni ceo broj	0	*Off*	Čekiran	Ne	Hex
Nevalidni ceo broj	Bilo koja druga	*On*	Nije čekiran	Da	Dec
Nevalidni ceo broj	Bilo koja druga	*Off*	Nije čekiran	Da	Dec
Alfabetski karakter	0	*On*	Nije čekiran	Ne	Dec
Alfabetski karakter	Bilo koja druga	*Off*	Čekiran	Da	Hex

Može se postaviti pitanje, da li je potrebno ići kroz sve ove duge korake sa svim ovim proverama i zamenama kada god nam je potrebna All-pairs analiza? Da li postoji način da se izvrši automatizacija ovog proračuna?

James Bach's All-Pairs Tool

James Bach kreirao je all-pair alat. Alat je dostupan na adresi www.satisfice.com. Pored samog alata, dostupno je upustvo, primeri i kod na Perl jeziku. Pogledajmo ukratko upotrebu ovog alata, na realnom primeru.

Recimo da testiramo online aplikaciju koja predstavlja sistem za rad sa hipotekama. Upotrebnom web čitača, korisnik dolazi na sajt, unosi lične podatke u niz formi. Sistem obradjuje podatke i u skladu sa podacima unetim i logikom programa u samoj aplikaciji, sistem govori korisniku koje za koje je hipotekarne kredite korisnik pogodan. Naredna slika prikazuje koje promenljive ova aplikacija može posedovati:

Region[1]	Tier[2]	Property	Credit[3]	Residence[4]	LTV[5]	NIV[6]	NAV[7]	Refinance	CC[8]	Intro Rate[9]	Emp. D[10]
NY	L	1 fam	A+	Pri	80%	Yes	Yes	Yes	Cust	Yes	Yes
NJ	M	2 fam	A	Vac	90%	No	No	No	Bank	No	No
FL	H	3 fam	A-	Inv	100%						
TX	H+1	4 fam	B								
CA	H+2	Coop	<B								
DC	H+3	Condo									
Other											

Imamo 7x6x6x5x3x3x2x2x2x2x2x2 = 725,760 validnih ulaza. All-pairs redukuje na 50.

In-Parameter-Order (IPO) procedure

Teorija (Osnove):

Procedura koju su predložili Yu Lei i Kuo-Chung Tai (HASE 1998, The 3rd IEEE International Symposium on High-Assurance Systems Engineering) za generisanje pokrivajućih nizova sa mešovitim nivoima. Procedura je poznata pod nazivom In-parameter Order (IPO) procedure.

Ulaz:
(a) n >= 2: Broj parametara (faktora);
(b) Broj vrednosti (nivoa) za svaki faktor.

Izlaz: MCA (pokrivajući niz sa mešovitim nivoima).

IPO procedura sastoji se iz tri koraka:
Korak 1: Glavna Procedura
Korak 2: Horizontalni Rast
Korak 3: Vertikalni Rast

Zadatak 1:
Razmotrimo program sa tri faktora A, B i C. A može uzeti vrednosti iz skupa $\{a_1, a_2, a_3\}$, B iz skupa $\{b_1, b_2\}$, i C iz skupa $\{c_1, c_2, c_3\}$. Potrebno je izgenerisati pokrivajući niz sa mešovitim nivoima, za tri navedena faktora.

Rešenje:
Rešenje započinjemo primenom Glavne Procedure, što je prvi korak za generisanje MCA upotrebom IPO algoritma.

Glavna Procedura (korak 1): Konstruišemo sve kombinacije parova vrednosti za prva dva faktora. Dobijamo sledeći skup parova:

$$T = \{(a_1, b_1), (a_1, b_2), (a_2, b_1), (a_2, b_2), (a_3, b_1), (a_3, b_2)\}$$

Elemente skupa *T* označićemo sa: $t_1, t_2, ..., t_6$.

Kompletna IPO procedura bi se završila u ovom trenutku, ukoliko bi broj parametara bio 2 (n=2). U našem primeru n=3, iz tog razloga potrebno je da nastavimo algoritam, korak koji sledi je Horizontalni Rast.

Horizontalni Rast (korak 1): Kreirati skup svih parova, AP, izmedju parametara A i C, zatim B i C. Ovo nas dovodi do narednog skupa od 15 parova:

$$AP=\{(a_1, c_1), (a_1, c_2), (a_1, c_3), (a_2, c_1), (a_2, c_2), (a_2, c_3), (a_3, c_1), (a_3, c_2), (a_3, c_3)$$
$$(b_1, c_1), (b_1, c_2), (b_1, c_3), (b_2, c_1), (b_2, c_2), (b_2, c_3)\}$$

Horizontalni Rast (korak 2): AP je skup parova koji još uvek nisu pokriveni. Označimo sa T' skup kombinacija dobijen proširenjem kombinacija iz T. U ovom trenutku T' je prazan, iz razloga što nismo do sada izvršili nikakva proširenja skupa T.

Horizontalni Rast (koraci 3, 4): Proširiti t1, t2, t3 dodavanjem c_1, c_2, c_3 vrednosti. Ovim proširenjem dobijamo:

t_1'=(a_1, b_1, c_1), t_2'=(a_1, b_2, c_2), i t_3'=(a_2, b_1, c_3)

Vršimo ažuriranje skupa T', koji sada postaje:

{a1, b1, c1), (a1, b2, c2), (a2, b1, c3)}

Ažuriramo skup parova koje je potrebno da pokrijemo

AP={(a_1, c_3), (a2, c1), (a2, c2), (a3, c1), (a3, c2), (a3, c3),
(b1, c2), (b2, c1), (b2, c3)}

Horizontalni Rast (korak 5): Još uvek nismo izvršili proširenja t4, t5, t6 jer C ne poseduje dovoljan broj elemenata. Najbolji način za proširenje ovih parova pronaćićemo u narednom koraku.

Horizontalni Rast (korak 6): Proširenje t4, t5, t6 pogodnim odabiranjem vrednosti C.

Ukoliko izvršimo proširenje t_4=(a_2, b_2) sa c_1, izvršićemo pokrivanje dva ne pokrivena para iz *AP* skupa, i to (a_2, c_1) i (b_2, c_1). Ukoliko proširenje izvršimo sa vrednošću c2, pokrivano samo jedan par iz skupa AP. Proširenjem sa c3, takodje pokrivano samo jedan par iz *AP* skupa. Stoga za proširenje t4 koristimo vrednost c1.

T'={(a_1, b_1, c_1), (a_1, b_2, c_2), (a_2, b_1, c_3), (a_2, b_2, c_1)}

AP= {(a_1, c_3), (a_2, c_2), (a_3, c_1), (a_3, c_2), (a_3, c_3), (b_1, c_2), (b_2, c_3)}

Slično, vršimo proširenje za t_5 i t_6, sa najboljim mogićim izborom vrednosti parametra C. Nakon odabira dobijamo:

t_5'=(a_3, b_1, c_3) and t_6'=(a_3, b_2, c_1)
t_5'=(a_3, b_1, c_2) and t_6'=(a_3, b_2, c_3)

T'={(a1, b1, c1), (a1, b2, c2), (a2, b1, c3), (a2, b2, c1), (a3, b1, c3), (a3, b2, c1)}
AP= {(a1, c3), (a2, c2), (a3, c2), (b1, c2), (b2, c3)}

Prethodnim korakom, završili smo horizontalni rast. Ono što treba primetiti jeste da je ostalo pet parova koji nisu pokriveni. To su:

AP= {(a1, c3), (a2, c2), (a3, c2), (b1, c2), (b2, c3)}

Pored toga izgenerisali smo šest kompletnih kombinacija:

T'={(a1, b1, c1), (a1, b2, c2), (a2, b1, c3), (a2, b2, c1), (a3, b1, c3), (a3, b2, c1)}

Sada prelazimo na Vertikalni Rast.

Vertikalni Rast: Za svaki par p, iz skupa *AP*, koji nedostaje, izgenerisaćemo novu kombinaciju u skupu T', tako da p bude pokriveno.
Počnimo sa parom p=(a_1, c_3).

Kombinacija t=(a_1, *, c_3) pokriva par p. Primetiti da vrednost za y nije bitna, znak * upravo predstavlja proizvoljnu vrednost.

Sledeće, razmotrimo p=(a_2, c_2). Ovaj par pokriven je kombinacijom (a_2, *, c_2)

Sledeće, razmotrimo p=(a_3, c_2). Ovaj par pokriven je kombinacijom (a_3, *, c_2)

Nadalje, razmotrimo p=(b_2, c_3). Već imamo kombinaciju (a1, *, c3), stoga možemo izvršiti modifikovanje iste u kombinaciju (a1, b2, c3). Tako da smo p pokrili bez dodavanja novih kombinacija.

Konačno, razmotrimo p=(b1, c2). Od ranije imamo kombinaciju (a3, *, c2), koju možemo modifikovati da bi dobili kombinaciju (a3, b1, c2). Ovom izmenom pokrili smo par p, bez dodavanja novih kombinacija.

Svaku, nebitnu vrednost (*), zamenjujemo proizvoljnom vrednošću odgovarajućeg faktora na osnovu čega dobijamo:

T={(a1, b1, c1), (a1, b2, c2), (a1, b1, c3), (a2, b1, c2), (a2, b2, c1),
(a2, b2, c3), (a3, b1, c3), (a3, b2, c1), (a3, b1, c2)}

Konačni pokrivajući niz:

MCA(9, 2^1, 3^2, 2)

Run	F1(X)	F2(Y)	F3(Z)
1	1	1	1
2	1	2	2
3	1	1	3
4	2	1	2
5	2	2	3
6	2	2	1
7	3	1	2
8	3	1	3
9	3	2	1

TESTIRANJE ZASNOVANO NA MODELU STANJA
(eng. State Testing)

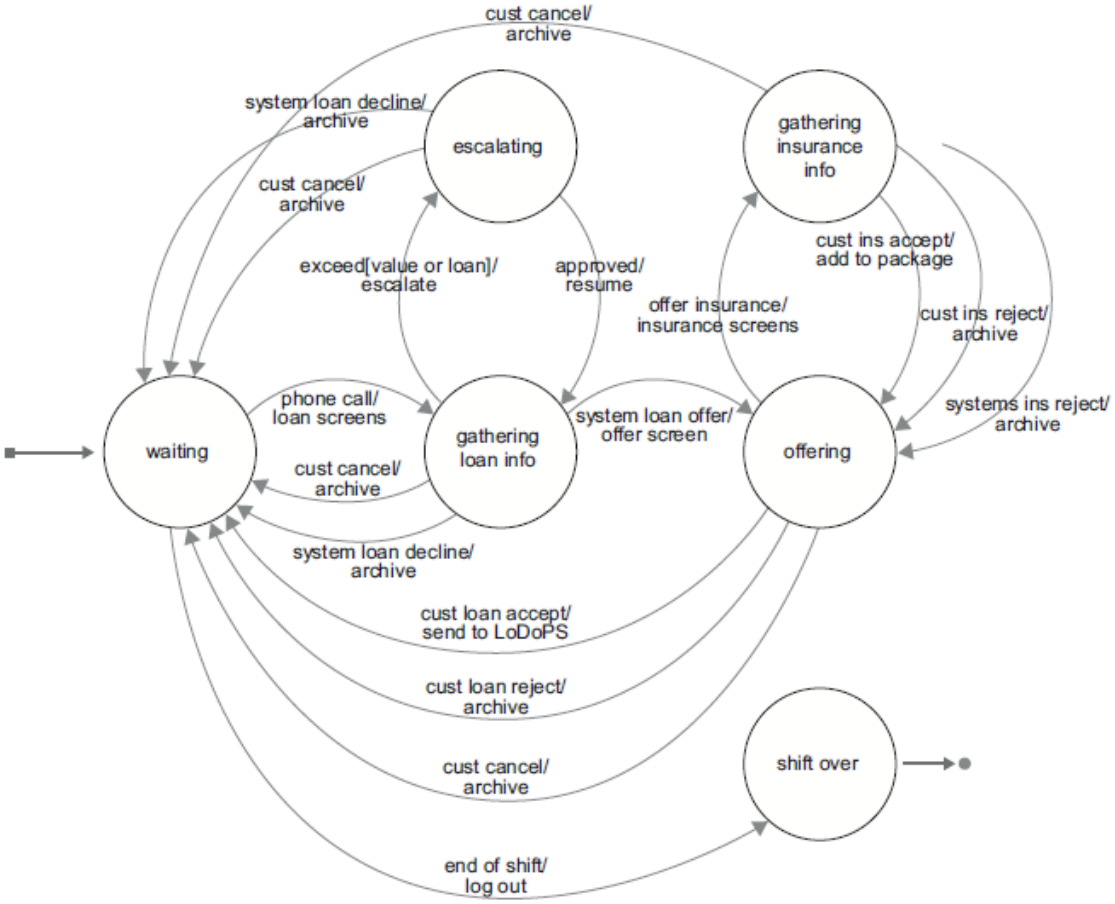
Zadatak 1:
Koristeći sledeću specifikaciju, prevesti je u dijagram stanja, prikazan iz ugla bankara. Generisati sve slučajeve testiranja koji će pokriti sva stanja i sve prelaze (0-switch). Formirati tabelu prelaza za 1-switch nivo.

Akter	Bankar
Preduslovi	Bankar jedne velike banke loguje se na sistem
Normalni tok	1. Bankar prihvata telefonski poziv od klijenta 2. Bankar ispituje klijenta, ubacuje informacije u sistem preko veb aplikacije 3. Kada bankar unese informacije o klijentu, sistem izračunava kreditnu sposobnost tog klijenta, koristeći deo aplikacije za bodovanje 4. Sistem, u zavisnosti od podataka o klijentu, prikazuje razne pakete koje bankar može ponuditi klijentu 5. Ako klijent odabere neki od tih paketa, bankar će uslovno potvrditi taj paket 6. Kada se intervjuisanje završi, bankar usmerava sistem da prosledi informacije o kreditu Sistemu za kreditnu sposobnost građana na odobravanje.
Alternativni tok 1	U koraku 2 normalnog toka, ako Klijent zahteva veliku sumu novca ili pozajmicu veću od vrednosti njegove imovine, bankar prebacuje apliciranje za kredit na senior bankara, koji dalje odlučuje o kreditu. Ako je odluka da se nastavi apliciranje, bankar završava ostatak aplikacije za kredit u koraku 2, i proces se nastavlja normalno. Ako je odluka da se ne nastavi apliciranje, bankar informiše klijenta da je aplikacija za kredit odbijena i intervju se završava.
Alternativni tok 2	U koraku 4 normalnog toka, ako sistem ne prikaže ni jedan paket koji je dostupan, bankar informiše klijenta da je aplikacija za kredit odbijena i intervju se završava.
Alternativni tok 3	Tokom koraka 5 normalnog toka, ako je klijent odabrao paket kratkoročnog zajma za stan, bankar nudi klijentu mogućnost podnošenja zahteva za životno osiguranje, da pokrije kredit. Ukoliko klijent želi to da prihvati, koraci su sledeći: 1) Bankar intervjuiše klijenta, unosti informacije o zdravlju u sistem koristeći veb aplikaciju. 2) Sistem obrađuje informacije i daje jedan od dva moguća izlaza: 2a) Sistem odbija da pruži osiguranje zasnovano na informacijama o zdravlju. Bankar informiše klijenta da je aplikacija za polisu osiguranja odbijena. Tok se završava i obrada se vraća u korak 5. 2b) Sistem nudi osiguranje po određenoj stopi na osnovu veličine kredita i informacijama o zdravlju klijenta. Bankar informiše klijenta o ponudi.

	3) Klijent može da odabere jednu od 2 odluke: 3a) Prihvata ponudu. Bankar vrši životno osiguranje klijenta kao deo aplikacije za kredit. Kada se završi ovaj alternativni tok, obrada se vraća u korak 5. 3b) Odbija ponudu. Bankar isključuje životno osigranje iz aplikacije za kredit. Kada se završi ovaj alternativni tok, obrada se vraća u korak 5.
Alternativni tok 4	U koracima od 1 do 5 normalnog toka, ako klijent u bulo kom trenutku odabere da završi intervju bez nastavka aplikaciranja za kredit ili odabira nekog paketa, apliciranje se prekida i intervju se završava.
Alternativni tok 5	Ako bankar nije logovan na sistem (npr. zato što je sistem pao) i počinje korak 1 normalnog toka, javiće se sledeći koraci: 1) Bankar unosi ručno informacije. Na kraju razgovora, bankar će informisati klijenta da će ga ponovo pozvati, kako bi mu saopštio odluku o apliciranju. 2) Kada se bankar ponovo prijavi na sistem, podaci iz aplikacije se unose u sistem i normalni tok se nastavlja na korak 2. 3) Bankar poziva klijenta ponovo da bi mu saopštio ishod: 3a) Dostigao se korak 5 normalnog toka i tok se nastavlja od tog koraka. 3b) U koraku 2 normalnog toka, desio se izuzetak toka 1. Tok se nastavlja u koraku 2. 3c) U koraku 4 normalnog toka, desio se izuzetak toka 2. Nema toka koji se nastavlja.
Posledice	Zahtev za kredit se šalje kreditnom birou u Los Anđelesu (LoDoPs) na odobravanje.

Rešenje:

Iz specifikacije zadatka možemo dobiti sledeći dijagram stanja:



Sada treba da generišemo slučajeve testiranja da pokrijemo sva stanja i prelaze (0-switch). Usvojićemo pravilo da svaki test mora početi u inicijalnom stanju (waiting) i mora se završiti u stanju čekanja (waiting) ili u stanju kraja smene, kada se bankar izloguje (shift out). Za postizanje pokrivenosti stanja i prelaza, dovoljno je izvršiti sledeće testove:

1. (waiting, phone call, loan screens, exceed[value], escale, approved, resume, system loan offer, offer screen, offer insurance, insurance screens, cust ins accept, add to package, cust loan accept, send to LoDoPS, waiting)

2. (waiting, phone call, loan screens, exceed[loan], escalate, approved, resume, system loan offer, offer screen, offer insurance, insurance sceens, cust ins reject, archive, cust loan accept, send to LoDoPS, waiting)

3. (waiting, phone call, loan screens, system loan offer, offer screen, offer insurance, insurance screens, system ins reject, archive, cust loan reject, archive, waiting)

4. (waiting, phone call, loan screens, exceed[loan], escalate, system loan decline, archivem waiting)

5. (waiting, phone call, loan screens, system loan decline, archive, waiting)

6. (waiting, phone call, loan screens, system loan decline, archive, waiting)

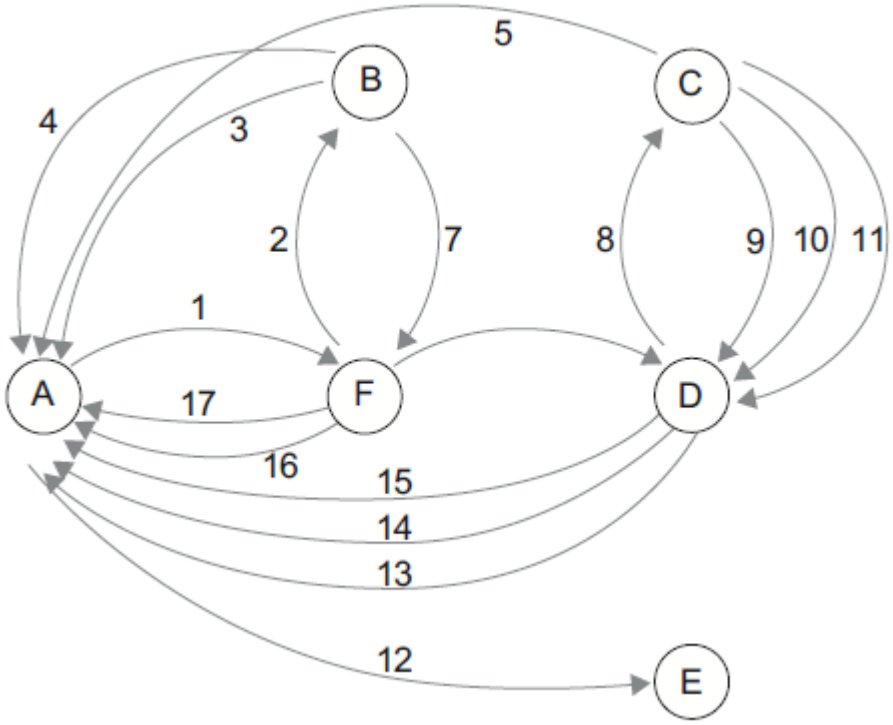
7. (waiting, phone call, loan screens, exceed[loan], escalate, cust cancel, archive, waiting)

8. (waiting, phone call, loan screens, system loan offer, offer screen, cust cancel, archive, waiting)

9. (waiting, phone call, loan screens, system loan offer, offer screen, offer insurance, insurance screens, cust cancel, archive, waiting)

10. (waiting, end of shift, log out, shift over)

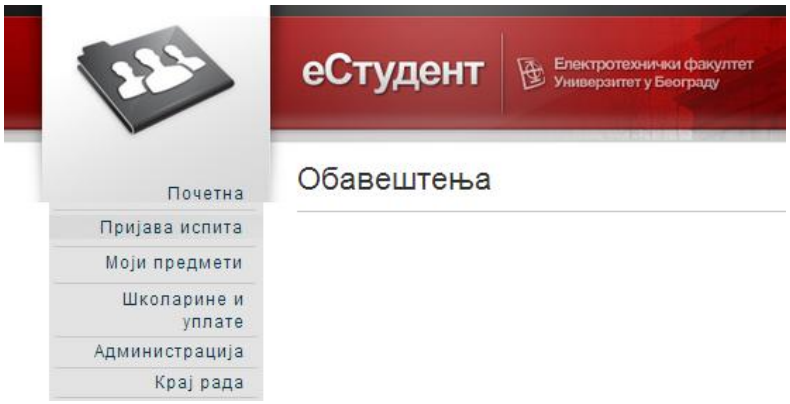
Dijagram stanja:



0-switch				1-switch			
A1	A12			A1F2	A1F6	A1F16	A1F17
B3	B4	B7		B3A1	B3A12		
				B4A1	B4A12		
				B7F2	B7F6	B7F16	B7F17
C5	C9	C10	C11	C5A1	C5A12		
				C9D8	C9D13	C9D14	C9D15
				C10D8	C10D13	C10D14	C10D15
				C11D8	C11D13	C11D14	C11D15
D8	D13	D14	D15	D8C5	D8C9	D8C10	D8C11
				D13A1	D13A12		
				D14A1	D14A12		
				D15A1	D15A12		
F2	F6	F16	F17	F2B3	F2B4	F2B7	
				F6D8	F6D13	F6D14	F6D15
				F16A1	F16A12		
				F17A1	F17A12		

Zadatak 2:

Dat je deo fakultetskog sistema e-Student. Da bi se ulogovao na sistem i mogao da izvrši određene funkcionalnosti, student mora imati nalog (korisničko ime i lozinku). Ukoliko unese ispravno korisničko ime i lozinku, student se uspešno loguje na sistem i dobija prvu stranicu sa obaveštenjima. U slučaju da unese pogrešno korisničko ime ili pogrešnu lozinku, student ostaje na stranici za logovanje. Na prvoj stranici, student vidi meni sa 5 mogućih opcija: Prijava ispita, Moji predmeti, Školarine i uplate, Administracija i Kraj rada. Kada je na stranici sa obaveštenjima, student može ići na neku od tih veb stranica.



Stranica „Prijava ispita“ sadrži spisak svih predmeta koje je student slušao i nije položio. Klikom na određeni predmet, student može da prijavi ispit i ukoliko uspešno prijavi, može da nastavi rad na istoj stranici, prijavom drugih predmeta. Ukoliko ne može da prijavi, to znači da je prekoračio limit u broju prijava. Prve tri prijave ispita nekog predmeta su besplatne, a ukoliko se ispit prijavljuje više od tri puta, tada student mora da uplati prijavu ispita, tako što će ga sistem prebaciti na stranicu „Školarine i uplate“. Takođe, ukoliko student pokuša da prijavi ispit, a prijava ispita nije u toku, dobiće obaveštenje o grešci „Prijava ispita nije u toku“ i biće vraćen na prvu stranicu sa obaveštenjima.

Druga opcija u meniju je „Moji predmeti“. Kada korisnik odabere tu opciju, dobija spisak svih predmeta koje sluša i koje je slušao, bez obzira da li ih je položio ili nije (kod onih predmeta koje je položio prikazuje se i ocena pored informacija o tom predmetu). Sa te stranice, ukoliko traje prijava predmeta na početku semestra, korisnik može da ide na stranicu „Biranje predmeta“. Tada on čekira predmete koje može izabrati i vraća se na stranicu „Moji predmeti“.

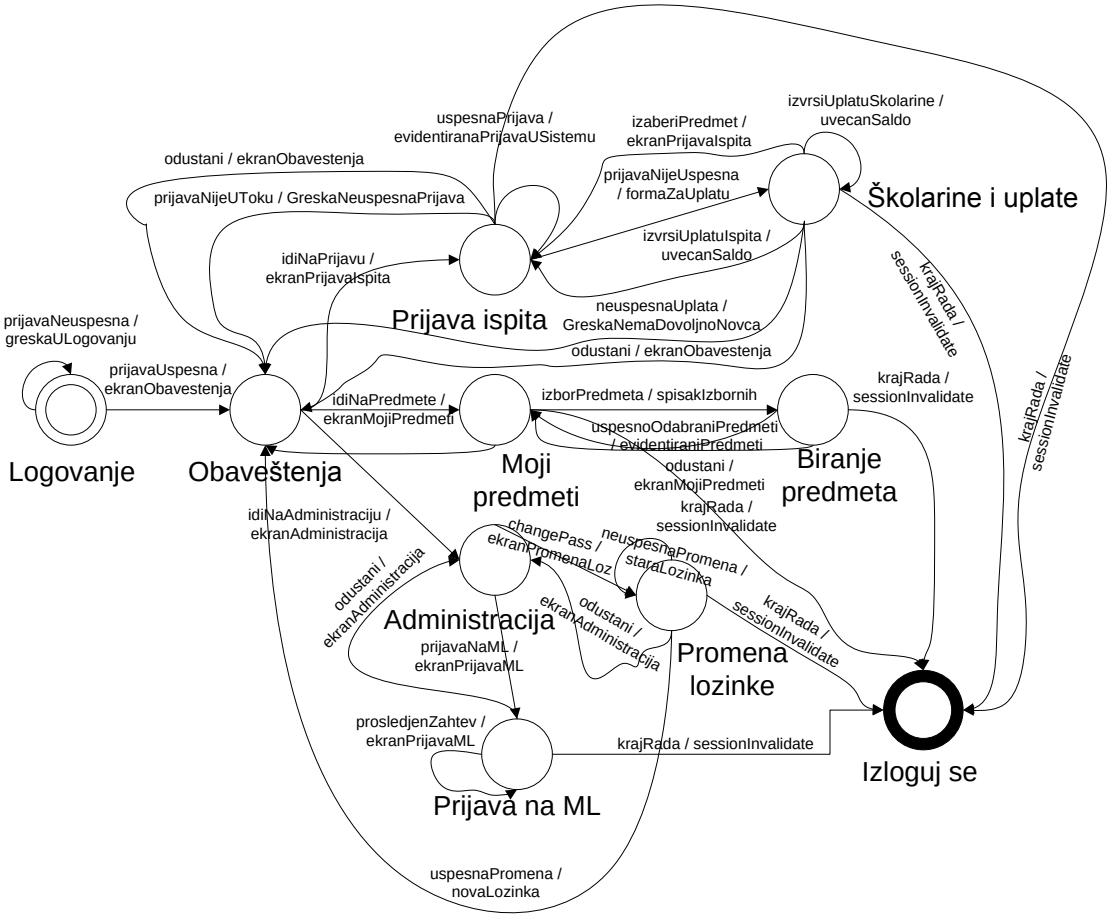
Opcija „Školarine i uplate“ daje pregled izvršenih školarina i uplata ispita. Student na tom ekranu može sa svog bankovnog računa da prebaci novac fakultetu ili za školarinu ili za uplatu ispita. Školarina se može uplatiti direktno sa tog ekrana, a ispit ne može, već se mora prvo odabrati koji ispit student želi da prijavi sa stranice „Prijava ispita“. Uplate može izvršiti samo ukoliko ima dovoljno novca na računu. Ukoliko nema dovoljno novca, izvršavanje akcije se obustavlja i vraća se na prvi ekran „Obaveštenja“ sa greškom „Nema dovoljno novca na računu“.

Stranica „Administracija“ ima 2 podopcije: „Promena lozinke“ i „Prijava na mejling listu“. Na ekranu „Promena lozinke“ student unosi novu lozinku i potvrđuje je. Ukoliko lozinka nema minimalno 5 slovnih karaktera i 3 numerička karaktera ili je identična kao prethodna lozinka, onda ostaje da važi stara lozinka i korisnik se vraća na stranicu „Administracija“. Ukoliko je uspešno uneta nova lozinka, korisnik se vraća na stranicu „Obaveštenja“ uz poruku sa servera „Nova lozinka je postavljena uspešno“. Opcijom „Prijava na mejling listu“ student može da za predmete koje sluša odluči da dobija obaveštenja putem mejla, pri čemu se šalje zahtev predmetnom nastavniku (i čeka se na odobrenje).

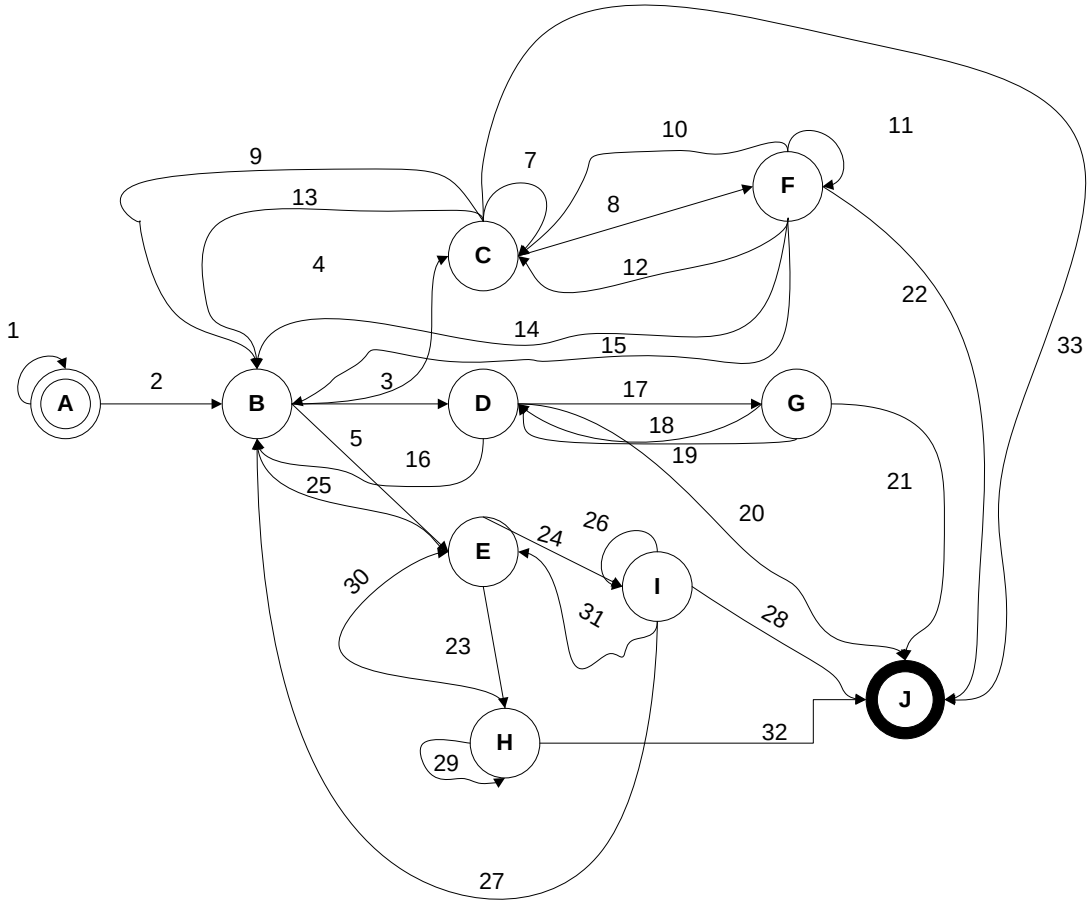
Sa bilo koje stranice ukoliko korisnik odluči da ide korak nazad i vraća se u roditeljsku stranicu (u hijerarhiji 0-nivo stranica je Logovanje, 1-nivo Obaveštenja, 2-nivo su stranice do kojih se dolazi iz menija, 3-nivo su podopcije tih stranica,... itd.). Takođe, u bilo kom trenutku korisnik može da se odluči za kraj rada u sistemu opcijom „Kraj rada“, kada se sesija sa serverom prekida, a student se upućuje na završnu stranicu na kojoj je izlogovan. Stavke glavnog menija korisnik vidi samo sa stranice „Obaveštenja“, odnosno te stranice nisu uvezane sa svim stranicama sistema.

Prevesti datu specifikaciju u dijagram stanja, prikazan iz ugla studenta. Kao akciju navoditi odgovor sistema, odnosno u slučaju ove veb aplikacije odgovor servera klijentu. Formirati tabelu za 0-switch i 1-switch.

Rešenje:


Logovanje
Obaveštenja
Prijava ispita
Moji predmeti
Biranje predmeta
Školarine i uplate
Administracija
Promena lozinke
Prijava na ML
Izloguj se
prijavaNeuspesna / greskaULogovanju
prijavaUspesna / ekranObavestenja
idiNaPrijavu / ekranPrijavaIspita
uspesnaPrijava / evidentiranaPrijavaUSistemu
odustani / ekranObavestenja
prijavaNijeUToku / GreskaNeuspesnaPrijava
neuspesnaUplata / GreskaNemaDovoljnoNovca
idiNaPredmete / ekranMojiPredmeti
izborPredmeta / spisakIzbornih
uspesnoOdabraniPredmeti / evidentiraniPredmeti
odustani / ekranMojiPredmeti
izaberiPredmet / ekranPrijavaIspita
prijavaNijeUspesna / formaZaUplatu
izvrsiUplatuIspita / uvecanSaldo
izvrsiUplatuSkolarine / uvecanSaldo
krajRada / sessionInvalidate
idiNaAdministraciju / ekranAdministracija
odustani / ekranAdministracija
changePass / ekranPromenaLoz
neuspesnaPromena / staraLozinka
prijavaNaML / ekranPrijavaML
prosledjenZahtev / ekranPrijavaML
uspesnaPromena / novaLozinka

Transformacijom datog dijagrama dobijamo:



0-switch						1-switch
A1	A2					A1A2, A2B2, A2B3, A2B5, A2B6
B3	B4	B5	B6			B3D16, B3D17, B3D20, B4C7, B4C8, B4C9, B4C13, B4C33, B5E24, B5E24, B5E25
C7	C8	C9	C13	C33		C7C7, C7C8, C7C9,C7C13, C7C33, C8F10, C8F11, C8F12, C8F14, C8F15, C8F22, C9B3, C9B4, C9B5, C9B6, C13B3, C13B4, C13B5, C13B6
D16	D17	D20				D16B3, D16B4, D16B5, D16B6, D17G18, D17G19, D17G21
E23	E24	E25				E23H29, E23H30, E23H32, E24I26, E24I27, E24I28, E24I31, E25B3, E25B4, E25B5, E25B6
F10	F11	F12	F14	F15	F22	F10C7, F10C8, F10C9, F10C13, F10C33, F11F10, F11F11, F11F12, F11F14, F11F15, F11F22, F12C7, F12C8, F12C9, F12C13, F12C33, F14B3, F14B4, F14B5, F14B6, F15B3, F15B4, F15B5, F15B6
G18	G19	G21				G18D16, G18D17, G18D20, G19D16, G19D17, G19D20

H29	H30	H32				H29H29, H29H30, H29H32, H30E23, H30E24, H30E25
I26	I27	I28	I31			I26I26, I26I27, I26I28, I26I31, I27B3, I27B4, I27B5, I27B6, I31E23, I31E24, I31E25