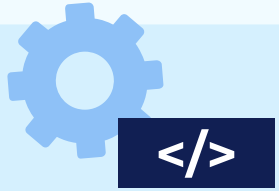
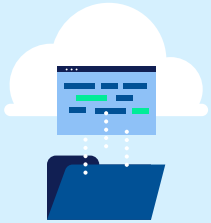
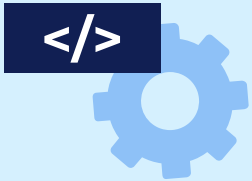
UNIVERZITET U BEOGRADU
ELEKTROTEHNIČKI
FAKULTET



MUTACIONO TESTIRANJE

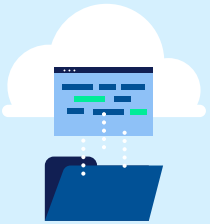


TESTIRANJE SOFTVERA



01

MUTACIJE PROGRAMA



MUTACIONO TESTIRANJE

- Mutaciono testiranje je tehnika koja se primarno koristi za procenu kvaliteta već napisanih test primera.
- Podrazumeva uvođenje mutacija u program:
  - **Mutacija** je mala izmena programa i predstavlja veštačku grešku umetnutu u njegov izvorni kod (npr. promena operatora ili konstante).
  - **Mutant** je program izveden iz nekog drugog programa mutacijom.
  - Mutant program mora da bude sintaksno korektan.
- Ukoliko je test primer u stanju da otkrije mutant program, odnosno da otkrije da se ponašanje razlikuje u odnosu na ponašanje originalnog programa, smatra se da je test primer dobro napisan.



MUTACIJE PROGRAMA

- **Mutant prvog reda** predstavlja modifikaciju originalnog programa, sa jednom mutacijom.
- **Mutant višeg reda** predstavlja modifikaciju originalnog programa, sa više mutacija.
  - Dobija se kombinacijom mutanata prvog reda.
- Nakon izvršavanja testova, a u zavisnosti od ishoda, mutant se klasifikuje kao:
  - **Ubijeni mutant** (*killed*) – ako ga je barem jedan test primer detektovao.
  - **Preživeli mutant** (*survived*) – ako ga nijedan test primer nije detektovao.
  - **Ekvivalentni mutant** (*equivalent*) – ako ne postoji test primer koji može da ga detektuje, odnosno on je semantički identičan originalnom programu.
  - Ukupan broj mutanata jednak je zbiru broja ubijenih, preživelih i ekvivalentnih mutanata.

POSTUPAK TESTIRANJA

1. Svaki test primer se izvršava nad originalnim programom i nad svakim generisanim mutant programom.
2. Za svaki par mutanta i test primera se beleži da li je mutant preživeo taj test primer ili je ubijen tim test primerom (*survived* ili *killed*).
3. Na osnovu njihovog statusa u testovima, određuje se konačan ishod svakog mutanta, a za svakog preživelog mutanta se utvrđuje da li je on ekvivalentan mutant.
4. Na osnovu ukupnog broja mutanata (**T**), broja ubijenih mutanata (**K**), broja preživelih mutanata (**S**) i broja ekvivalentnih mutanata (**E**) određuje se **mutacioni skor**, koji može biti između 0 i 1, na jedan od sledeća dva načina:

$$MS(T) = K / (T - E)$$

ili

$$MS(T) = K / (K + S)$$

MUTACIONA MATRICA

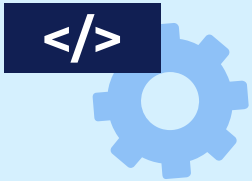
- Mutaciona matrica sadrži status svakog mutant programa nakon izvršavanja svakog test primera.
  - U vrstama se najčešće navode mutanti sa opisom primenjene mutacije.
  - U kolonama se najčešće navode test primeri.
  - Poslednja kolona sadrži konačni ishod mutanta nakon svih izvršenih testova.
  - Mogu se zameniti vrste i kolone.
- Primer mutacione matrice za tri mutant programa i skup od tri test primera:

M	Opis mutanta	TP_1	TP_2	TP_3	Ishod mutanta
M_1	`Linija: Mutacija`	S	K	S	Ubijen
M_2	`Linija: Mutacija`	S	S	S	Preživeo
M_3	`Linija: Mutacija`	S	S	K	Ubijen

TIPOVI MUTACIJA

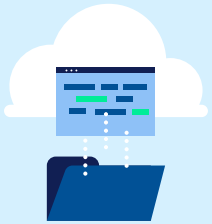
- **Jake mutacije** proveravaju isključivo da li se ponašanje originalnog i mutant programa razlikuje u verifikacionim tačkama.
  - Na primer, samo se posmatra da li je povratna vrednost funkcije ista ili različita.
  - Mutacija mora da se propagira do verifikacionih tačaka.
- **Slabe mutacije** proveravaju da li se u bilo kom trenutku tokom izvršavanja originalnog i mutant programa stanje ili ponašanje razlikovalo.
  - Na primer, posmatra se vrednost svake memorijske lokacije i registra.
  - Mutacija ne mora da se propagira do verifikacionih tačaka.
- Mutacije su podrazumevano jake.





02

ZADACI



ZADATAK 1 – SABERI ILI POMNOŽI

POSTAVKA (1)

- Data je funkcija **ADD_OR_MUL** u pseudokodu koja određuje ili zbir ili proizvod dva prosleđena cela broja i vraća odgovarajući rezultat.
- Takođe, dat je skup test primera **T** koji je iskorišćen za testiranje ove funkcije.

```
ADD_OR_MUL(x, y)
1. if (x < y) then
2.     return x + y
3. else
4.     return x * y
5. end_if
```

TP	x	y
1	0	0
2	0	1
3	1	0
4	-1	-2

ZADATAK 1 – SABERI ILI POMNOŽI

POSTAVKA (2)

1. Odrediti sve mutante prvog reda date funkcije ukoliko se koriste sledeće mutacije:
   - Operator **+** se menja operatorom **–**.
   - Operator ***** se menja operatorom **/**.
   - Celobrojna promenljiva **p** se menja izrazom **p+1**.
2. Odrediti izlaz generisanih mutanata za svaki test primer iz skupa T. Konstruisati mutacionu matricu za skup test primera T. Za svakog mutanta odrediti i njegov konačni ishod nakon izvršavanja svih testova.
3. Odrediti broj ekvivalentnih mutanata. Ukoliko ne postoji nijedan ekvivalentni mutant, dati primer jedne ekvivalentne mutant funkcije.
4. Odrediti mutacioni skor datog skupa test primera.

ZADATAK 1 – SABERI ILI POMNOŽI

REŠENJE – MUTANTI (1)

- U prvoj liniji funkcije je moguće primeniti samo mutaciju promenljive. Dobijaju se dve mutant funkcije prvog reda:

```
ADD_OR_MUL(x, y)
1. if (x < y) then
2.     return x + y
3. else
4.     return x * y
5. end_if
```

Originalni program

```
ADD_OR_MUL(x, y)
1. if ((x+1) < y) then
2.     return x + y
3. else
4.     return x * y
5. end_if
```

Mutant M_1

```
ADD_OR_MUL(x, y)
1. if (x < (y+1)) then
2.     return x + y
3. else
4.     return x * y
5. end_if
```

Mutant M_2



ZADATAK 1 – SABERI ILI POMNOŽI

REŠENJE – MUTANTI (2)

- U drugoj liniji funkcije je moguće primeniti mutaciju promenljive i mutaciju „+ u –“. Dobijaju se tri mutant funkcije prvog reda:

```
ADD_OR_MUL(x, y)
1. if (x < y) then
2.     return x + y
3. else
4.     return x * y
5. end_if
```

Originalni program

```
ADD_OR_MUL(x, y)
1. if (x < y) then
2.     return (x+1) + y
3. else
4.     return x * y
5. end_if
```

Mutant M_3

```
ADD_OR_MUL(x, y)
1. if (x < y) then
2.     return x + (y+1)
3. else
4.     return x * y
5. end_if
```

Mutant M_4

```
ADD_OR_MUL(x, y)
1. if (x < y) then
2.     return x - y
3. else
4.     return x * y
5. end_if
```

Mutant M_5

ZADATAK 1 – SABERI ILI POMNOŽI

REŠENJE – MUTANTI (3)

- U četvrtoj liniji funkcije je moguće primeniti mutaciju promenljive i mutaciju „* u /“. Dobijaju se tri mutant funkcije prvog reda:

```
ADD_OR_MUL(x, y)
1.  if (x < y) then
2.      return x + y
3.  else
4.      return x * y
5.  end_if
```

Originalni program

```
ADD_OR_MUL(x, y)
1.  if (x < y) then
2.      return x + y
3.  else
4.      return (x+1) * y
5.  end_if
```

Mutant M_6

```
ADD_OR_MUL(x, y)
1.  if (x < y) then
2.      return x + y
3.  else
4.      return x * (y+1)
5.  end_if
```

Mutant M_7

```
ADD_OR_MUL(x, y)
1.  if (x < y) then
2.      return x + y
3.  else
4.      return x / y
5.  end_if
```

Mutant M_8

ZADATAK 1 – SABERI ILI POMNOŽI

REŠENJE – MUTACIONA MATRICA (1)

- Prvi korak je određivanje izlaza svake mutant funkcije za date test primere i utvrđivanje da li se ti izlazi razlikuju u odnosu na izlaz originalnog programa:

M	Opis mutanta	TP_1	TP_2	TP_3	TP_4
O	-	0	1	0	2
M_1	`1: x < y → (x+1) < y`	0	0	0	2
M_2	`1: x < y → x < (y+1)`	0	1	0	2
M_3	`2: x + y → (x+1) + y`	0	2	0	2
M_4	`2: x + y → x + (y+1)`	0	2	0	2
M_5	`2: x + y → x - y`	0	-1	0	2
M_6	`4: x * y → (x+1) * y`	0	1	0	0
M_7	`4: x * y → x * (y+1)`	0	1	1	1
M_8	`4: x * y → x / y`	undefined	1	undefined	0.5

```
ADD_OR_MUL(x, y)
1.  if (x < y) then
2.      return x + y
3.  else
4.      return x * y
5.  end_if
```

TP	x	y
1	0	0
2	0	1
3	1	0
4	-1	-2

ZADATAK 1 – SABERI ILI POMNOŽI

REŠENJE – MUTACIONA MATRICA (2)

- U mutacionu matricu se upisuje **K** (*killed*) ukoliko je mutant ubijen test primerom, a **S** (*survived*) ukoliko je mutant preživeo test primer.

M	Opis mutanta	TP_1	TP_2	TP_3	TP_4
M_1	`1: x < y → (x+1) < y`	S	K	S	S
M_2	`1: x < y → x < (y+1)`	S	S	S	S
M_3	`2: x + y → (x+1) + y`	S	K	S	S
M_4	`2: x + y → x + (y+1)`	S	K	S	S
M_5	`2: x + y → x - y`	S	K	S	S
M_6	`4: x * y → (x+1) * y`	S	S	S	K
M_7	`4: x * y → x * (y+1)`	S	S	K	K
M_8	`4: x * y → x / y`	K	S	K	K

ZADATAK 1 – SABERI ILI POMNOŽI

REŠENJE – MUTACIONA MATRICA (3)

- Ukoliko je mutanta ubio barem jedan test primer, njegov konačan ishod je da je ubijen, a ukoliko je preživeo sve test primere, njegov konačan ishod je da je preživeo.

M	Opis mutanta	TP_1	TP_2	TP_3	TP_4	Ishod mutanta
M_1	`1: x < y → (x+1) < y`	S	K	S	S	Ubijen
M_2	`1: x < y → x < (y+1)`	S	S	S	S	Preživeo
M_3	`2: x + y → (x+1) + y`	S	K	S	S	Ubijen
M_4	`2: x + y → x + (y+1)`	S	K	S	S	Ubijen
M_5	`2: x + y → x - y`	S	K	S	S	Ubijen
M_6	`4: x * y → (x+1) * y`	S	S	S	K	Ubijen
M_7	`4: x * y → x * (y+1)`	S	S	K	K	Ubijen
M_8	`4: x * y → x / y`	K	S	K	K	Ubijen

ZADATAK 1 – SABERI ILI POMNOŽI

REŠENJE – EKVIVALENTNI MUTANTI (1)

- Za preživele mutante se mora proveriti da li su oni ekvivalentni mutanti.
- Jedini preživeli mutant je M_2.

M	Opis mutanta	TP_1	TP_2	TP_3	TP_4	Ishod mutanta
M_2	`1: x < y → x < (y+1)`	S	S	S	S	Preživeo

- Da li postoji neki test primer koji bi ubio ovog mutanta?
  - Na primer: x = 3, y = 3.
  - Izlaz originalnog programa bi bio 9, pošto uslov 3 < 3 nije ispunjen.
  - Izlaz mutanta M_2 bi bio 6, pošto je uslov 3 < 4 ispunjen.
- Dakle, mutant M_2 nije ekvivalentni mutant, tako da je broj ekvivalentnih mutanata 0.

ZADATAK 1 – SABERI ILI POMNOŽI

REŠENJE – EKVIVALENTNI MUTANTI (2)

- Kako bi izgledao ekvivalentni mutant originalne funkcije?
  - Operacije sabiranja i množenja su komutativne, tako da zamena mesta operandima ne utiče na rezultat.
  - Može da se iskoristi mutacija zamene mesta operanada u liniji 2 ili liniji 4.
  - Na ovaj način se dobija semantički ekvivalentna funkcija.

```
ADD_OR_MUL(x, y)
1. if (x < y) then
2.     return y + x
3. else
4.     return x * y
5. end_if
```



ZADATAK 1 – SABERI ILI POMNOŽI

REŠENJE – MUTACIONI SKOR

- Ukupan broj mutanata je 8.
- Broj ubijenih mutanata je 7.
- Broj preživelih mutanata je 1.
- Broj ekvivalentnih mutanata je 0.

- Mutacioni skor skupa test primera T je:

$$MS(T) = 7 / (7 + 1) = 0.875$$

ili

$$MS(T) = 7 / (8 - 0) = 0.875$$

ZADATAK 2 – KREDIT

POSTAVKA (1)

- Data je implementacija funkcije **loanApproval** u programskom jeziku Java koja određuje da li je klijent banke kreditno sposoban. Funkcija prihvata godišnju platu klijenta izraženu u dolarima, a vraća logičku vrednost koja određuje da li mu je kredit odobren ili ne.
- Takođe, dat je skup test primera **T** koji je iskorišćen za testiranje ove funkcije.

```
public boolean loanApproval(double salary) {
1.  if (salary >= 30000) return true;
2.  else return false;
3. }
```

TP	salary
1	0
2	15000
3	35000



ZADATAK 2 – KREDIT

POSTAVKA (2)

1. Odrediti sve mutante prvog reda date funkcije ukoliko se koristi mutacija zamene relacionih operatora, kojom se jedan relacioni operator menja svakim drugim relacionim operatorom.
2. Odrediti izlaz generisanih mutanata za svaki test primer iz skupa T. Konstruisati mutacionu matricu za skup test primera T. Za svakog mutanta odrediti i njegov konačni ishod nakon izvršavanja svih testova.
3. Odrediti mutacioni skor datog skupa test primera.
4. Ukoliko mutacioni skor nije maksimalan, dopuniti skup test primera T minimalnim brojem test primera tako da se mutacioni skor maksimizuje.



ZADATAK 2 – KREDIT

REŠENJE – MUTANTI

- Relacioni operatori su: **>=**, **>**, **<=**, **<**, **==** i **!=**. Dakle, postoji pet mutanata prvog reda:

O

```
public boolean loanApproval(double salary) {
1.   if (salary >= 30000) return true;
2.   else return false;
3. }
```

M_1

```
public boolean loanApproval(double salary) {
1.   if (salary > 30000) return true;
2.   else return false;
3. }
```

M_2

```
public boolean loanApproval(double salary) {
1.   if (salary < 30000) return true;
2.   else return false;
3. }
```

M_3

```
public boolean loanApproval(double salary) {
1.   if (salary <= 30000) return true;
2.   else return false;
3. }
```

M_4

```
public boolean loanApproval(double salary) {
1.   if (salary == 30000) return true;
2.   else return false;
3. }
```

M_5

```
public boolean loanApproval(double salary) {
1.   if (salary != 30000) return true;
2.   else return false;
3. }
```

ZADATAK 2 – KREDIT

REŠENJE – MUTACIONA MATRICA (1)

- Izlazi originalnog programa i mutant funkcija za skup test primera T:

M	Opis mutanta	TP_1	TP_2	TP_3
O	-	false	false	true
M_1	1: >= → >	false	false	true
M_2	1: >= → <	true	true	false
M_3	1: >= → <=	true	true	false
M_4	1: >= → ==	false	false	false
M_5	1: >= → !=	true	true	true

```
public boolean
loanApproval(double salary) {
1.  if (salary >= 30000)
    return true;
2.  else return false;
3.  }
```

TP	salary
1	0
2	15000
3	35000

ZADATAK 2 – KREDIT

REŠENJE – MUTACIONA MATRICA (2)

- Mutaciona matrica za skup test primera T:

M	Opis mutanta	TP_1	TP_2	TP_3	Ishod mutanta
M_1	1: >= → >	S	S	S	Preživeo
M_2	1: >= → <	K	K	K	Ubijen
M_3	1: >= → <=	K	K	K	Ubijen
M_4	1: >= → ==	S	S	K	Ubijen
M_5	1: >= → !=	K	K	S	Ubijen

```
public boolean
loanApproval(double salary) {
1.  if (salary >= 30000)
    return true;
2.  else return false;
3.  }
```

TP	salary
1	0
2	15000
3	35000

ZADATAK 2 – KREDIT

REŠENJE – MUTACIONI SKOR (1)

- Jedini preživeli mutant je M_1. Da li je on ekvivalentni mutant?

M	Opis mutanta	TP_1	TP_2	TP_3	Ishod mutanta
M_1	`1: >= → >`	S	S	S	Preživeo

- Da li postoji neki test primer kojim bi se ubio ovaj mutant?
  - Na primer: salary = 30000.
  - Izlaz originalnog programa bi bio *true*, pošto je uslov ispunjen.
  - Izlaz mutanta M_1 bi bio *false*, pošto uslov nije ispunjen.
- Dakle, nijedan preživeli mutant nije ekvivalentni mutant.

ZADATAK 2 – KREDIT

REŠENJE – MUTACIONI SKOR (2)

- Ukupan broj mutanata je 5.
- Broj ubijenih mutanata je 4.
- Broj preživelih mutanata je 1.
- Broj ekvivalentnih mutanata je 0.

- Mutacioni skor skupa test primera T je: **MS(T) = 4 / (4 + 1) = 0.8**

- Da li je ovo maksimalni mogući mutacioni skor za korišćeni skup mutanata?
  - Pošto nema ekvivalentnih mutanata, mutacioni skor nije maksimizovan i treba ubiti sve preživele mutante.



ZADATAK 2 – KREDIT

REŠENJE – MUTACIONI SKOR (3)

- Za jedinog mutanta koji je preživeo je utvrđeno da nije ekvivalentni mutant, tako da postoji test primer koji može da ga ubije:
  - Na primer: salary = 30000.
- Mutaciona matrica i dopunjeni skup test primera **T'** sa kojim mutacioni skor dostiže 1:

M	Opis mutanta	TP_1	TP_2	TP_3	TP_4	Ishod mutanta
M_1	1: >= → >	S	S	S	K	Ubijen
M_2	1: >= → <	K	K	K	K	Ubijen
M_3	1: >= → <=	K	K	K	S	Ubijen
M_4	1: >= → ==	S	S	K	S	Ubijen
M_5	1: >= → !=	K	K	S	K	Ubijen

TP	salary
1	0
2	15000
3	35000
4	30000

ZADATAK 3 – CENA PAKETA

POSTAVKA (1)

- Data je implementacija funkcije **calculate_price** u programskom jeziku Python koja određuje cenu paketa u okviru jedne veb prodavnice. Funkcija prihvata originalnu cenu paketa u dolarima i celobrojnu vrednost koja predstavlja popust na osnovu primenjenog promotivnog koda, a vraća cenu paketa sa popustom izraženu, takođe, u dolarima.

```
def calculate_price(price, discount):
1.  if discount > 50:
2.      discount = 50
3.  if discount < 0:
4.      discount = 0
5.  price = price * (1 - discount / 100)
6.  return round(price, 2)
```

ZADATAK 3 – CENA PAKETA

POSTAVKA (2)

- Dodatno, date su i sledeće tri mutant funkcije prvog reda:

```
def calculate_price(price, discount):
1.    if discount < 50:
2.       discount = 50
3.    if discount < 0:
4.       discount = 0
5.    price = price * (1 - discount / 100)
6.    return round(price, 2)
```

Mutant M_1

```
def calculate_price(price, discount):
1.    if discount > 50:
2.       discount = 50
3.    if discount < 10:
4.       discount = 0
5.    price = price * (1 - discount / 100)
6.    return round(price, 2)
```

Mutant M_2

```
def calculate_price(price, discount):
1.    if discount > 50:
2.       discount = 50
3.    if discount < 0:
4.       discount = 1
5.    price = price * (1 - discount / 100)
6.    return round(price, 2)
```

Mutant M_3

ZADATAK 3 – CENA PAKETA

POSTAVKA (3)

1. Odrediti minimalan skup test primera T kojim će biti ubijeni svi dati mutanti prvog reda.
2. Odrediti sve mutante drugog reda kombinujući date mutante prvog reda.
3. Da li test primeri iz skupa T ubijaju i sve mutante drugog reda? Ako ne, dopuniti skup T minimalnim brojem test primera tako da i svi mutanti drugog reda budu ubijeni.



ZADATAK 3 – CENA PAKETA

REŠENJE – TEST PRIMERI

- Skup test primera T kojim će biti ubijene sve mutant funkcije:

TP	price	discount
1	100.00	-10
2	100.00	5

- Izlazi originalnog programa i mutant funkcija za skup test primera T:

M	Opis mutanta	TP_1	TP_2
O	-	100.00	95.00
M_1	`1: > → <`	50.00	50.00
M_2	`3: 0 → 10`	100.00	100.00
M_3	`4: 0 → 1`	99.00	95.00

ZADATAK 3 – CENA PAKETA

REŠENJE – MUTANTI PRVOG REDA

- Mutaciona matrica za skup test primera T i mutant funkcije prvog reda:

M	Opis mutanta	TP_1	TP_2	Ishod mutanta
M_1	`1: > → <`	K	K	Ubijen
M_2	`3: 0 → 10`	S	K	Ubijen
M_3	`4: 0 → 1`	K	S	Ubijen

ZADATAK 3 – CENA PAKETA

REŠENJE – MUTANTI DRUGOG REDA (1)

- Mutanti drugog reda originalne funkcije:

```
def calculate_price(price, discount):
1.    if discount < 50:
2.        discount = 50
3.    if discount < 10:
4.        discount = 0
5.    price = price * (1 - discount / 100)
6.    return round(price, 2)
```

Mutant $M_4 = M_1 + M_2$

```
def calculate_price(price, discount):
1.    if discount < 50:
2.        discount = 50
3.    if discount < 0:
4.        discount = 1
5.    price = price * (1 - discount / 100)
6.    return round(price, 2)
```

Mutant $M_5 = M_1 + M_3$

```
def calculate_price(price, discount):
1.    if discount > 50:
2.        discount = 50
3.    if discount < 10:
4.        discount = 1
5.    price = price * (1 - discount / 100)
6.    return round(price, 2)
```

Mutant $M_6 = M_2 + M_3$

ZADATAK 3 – CENA PAKETA

REŠENJE – MUTANTI DRUGOG REDA (2)

- Izlazi originalnog programa i mutant funkcija drugog reda za skup test primera T:

M	Opis mutanta	TP_1	TP_2
O	-	100.00	95.00
M_4	`1: > → <` `3: 0 → 10`	50.00	50.00
M_5	`1: > → <` `4: 0 → 1`	50.00	50.00
M_6	`3: 0 → 10` `4: 0 → 1`	99.00	99.00

- Postojeći test primeri ubijaju i sve mutante drugog reda.
- Da li ubijanje svih mutanata prvog reda garantuje ubijanje svih mutanata višeg reda koji se dobijaju kombinacijom mutanata prvog reda?

ZADATAK 4 – TROŠKOVI DOSTAVE

POSTAVKA (1)

- Data je implementacija funkcije **shipping_cost** u programskom jeziku Python koja određuje cenu dostave paketa u okviru jedne veb prodavnice. Funkcija prihvata težinu paketa u funtama i logičku vrednost koja označava da li je odabrana ekspresna dostava, a vraća ukupan trošak dostave izražen u dolarima.
- Takođe, dat je skup test primera **T** koji je iskorišćen za testiranje ove funkcije.

```
def shipping_cost(weight, express):
1.  cost = 5
2.  if weight > 10:
3.      cost += 4
4.  if express:
5.      cost = 12
6.  return cost
```

TP	weight	express
1	10	False
2	12	True
3	5	False

ZADATAK 4 – TROŠKOVI DOSTAVE

POSTAVKA (2)

- Dodatno, date su i sledeće tri mutant funkcije prvog reda:

```
def shipping_cost(weight, express):
1.   cost = 5
2.   if weight >= 10:
3.      cost += 4
4.   if express:
5.      cost = 12
6.   return cost
```

Mutant M_1

```
def shipping_cost(weight, express):
1.   cost = 5
2.   if weight > 10:
3.      cost += 5
4.   if express:
5.      cost = 12
6.   return cost
```

Mutant M_2

```
def shipping_cost(weight, express):
1.   cost = 5
2.   if weight > 10:
3.      cost += 4
4.   if not express:
5.      cost = 12
6.   return cost
```

Mutant M_3

ZADATAK 4 – TROŠKOVI DOSTAVE

POSTAVKA (3)

1. Odrediti izlaz datih mutant funkcija za svaki test primer iz skupa T i konstruisati mutacionu matricu za skup test primera T, ukoliko se koriste:
   a) Jake mutacije
   b) Slabe mutacije

   Za svakog mutanta odrediti i njegov konačni ishod nakon izvršavanja svih testova.
2. Navesti koji mutanti su ubijeni u oba slučaja, a koji u samo jednom slučaju.
3. Dopuniti skup test primera T minimalnim brojem test primera tako da svi mutanti budu ubijeni u oba slučaja mutacija.



ZADATAK 4 – TROŠKOVI DOSTAVE

REŠENJE – JAKE MUTACIJE (1)

- Izlazi originalnog programa i mutant funkcija za skup test primera T:

M	Opis mutanta	TP_1	TP_2	TP_3
O	-	5	12	5
M_1	2: > → >=	9	12	5
M_2	3: 4 → 5	5	12	5
M_3	4: express → not express	12	9	12

```
def shipping_cost(weight, express):
1.    cost = 5
2.    if weight > 10:
3.       cost += 4
4.    if express:
5.       cost = 12
6.    return cost
```

TP	weight	express
1	10	False
2	12	True
3	5	False

ZADATAK 4 – TROŠKOVI DOSTAVE

REŠENJE – JAKE MUTACIJE (2)

- Mutaciona matrica za skup test primera T ukoliko se koriste **jake** mutacije:

M	Opis mutanta	TP_1	TP_2	TP_3	Ishod mutanta
M_1	`2: > → >=`	K	S	S	Ubijen
M_2	`3: 4 → 5`	S	S	S	Preživeo
M_3	`4: express → not express`	K	K	K	Ubijen

```
def shipping_cost(weight, express):
1.    cost = 5
2.    if weight > 10:
3.       cost += 4
4.    if express:
5.       cost = 12
6.    return cost
```

TP	weight	express
1	10	False
2	12	True
3	5	False



ZADATAK 4 – TROŠKOVI DOSTAVE

REŠENJE – SLABE MUTACIJE (1)

- Slaba mutacija podrazumeva da će mutant biti ubijen ukoliko se u bilo kom trenutku stanje mutanta razlikuje u odnosu na stanje originalnog programa.
- Kako se ponaša mutant M_1 za drugi test primer?
  - Uslov na liniji 2 će biti zadovoljen i u originalnom programu i u mutantu, pa će u oba slučaja vrednost promenljive *cost* postati 9.
  - Uslov na liniji 4 će takođe biti zadovoljen u oba slučaja, pa će vrednost promenljive *cost* postati 12.
  - Povratna vrednost originalne funkcije i mutanta je ista i jednaka je 12.
  - Ni u jednom trenutku tokom izvršavanja testa se stanje nije razlikovalo za originalnu i mutant funkciju.
  - Dakle, mutant M_1 preživljava drugi test primer sa slabom mutacijom.



ZADATAK 4 – TROŠKOVI DOSTAVE

REŠENJE – SLABE MUTACIJE (2)

- Kako se ponaša mutant M_2 za drugi test primer?
  - Uslov na liniji 2 će biti zadovoljen i u originalnom programu i u mutantu, međutim, naredba koja se izvršava ako je uslov zadovoljen se razlikuje.
  - U slučaju originalne funkcije, vrednost promenljive *cost* će postati 9.
  - U slučaju mutant funkcije, vrednost promenljive *cost* će postati 10.
  - Uslov na liniji 4 će biti zadovoljen i u originalnom programu i u mutantu i vrednost promenljive *cost* će postati 12 u oba slučaja, što je i povratna vrednost funkcije.
  - Bez obzira na to, na prvom mestu gde se stanje (vrednost promenljive) razlikuje, izvršavanje testa se zaustavlja i smatra se da je mutant ubijen.
  - Dakle, mutanta M_2 ubija drugi test primer sa slabom mutacijom.



ZADATAK 4 – TROŠKOVI DOSTAVE

REŠENJE – SLABE MUTACIJE (3)

- Slično se mogu odrediti ishodi i za prvi i treći test primer.
- Mutaciona matrica za skup test primera T ukoliko se koriste **slabe** mutacije:

M	Opis mutanta	TP_1	TP_2	TP_3	Ishod mutanta
M_1	`2: > → >=`	K	S	S	Ubijen
M_2	`3: 4 → 5`	S	K	S	Ubijen
M_3	`4: express → not express`	K	K	K	Ubijen

```
def shipping_cost(weight, express):
1.    cost = 5
2.    if weight > 10:
3.       cost += 4
4.    if express:
5.       cost = 12
6.    return cost
```

TP	weight	express
1	10	False
2	12	True
3	5	False



ZADATAK 4 – TROŠKOVI DOSTAVE

REŠENJE – MUTACIONE MATRICE (1)

- Mutanti M_1 i M_3 su ubijeni i u slučaju jake i u slučaju slabe mutacije, dok je mutant M_2 ubijen samo u slučaju slabe mutacije.

M	Opis mutanta	TP_1	TP_2	TP_3	Ishod mutanta
M_1	`2: > → >=`	K	S	S	Ubijen
M_2	`3: 4 → 5`	S	S	S	Preživeo
M_3	`4: express → not express`	K	K	K	Ubijen

Jake mutacije

M	Opis mutanta	TP_1	TP_2	TP_3	Ishod mutanta
M_1	`2: > → >=`	K	S	S	Ubijen
M_2	`3: 4 → 5`	S	K	S	Ubijen
M_3	`4: express → not express`	K	K	K	Ubijen

Slabe mutacije

ZADATAK 4 – TROŠKOVI DOSTAVE

REŠENJE – MUTACIONE MATRICE (2)

- Da li je za mutante koji su ubijeni jakom mutacijom potrebno proveravati da li preživljavaju slabu mutaciju?
  - Ne, jer ubijanje jakom mutacijom garantuje i ubijanje slabom mutacijom.
  - Ako se povratna vrednost mutant funkcije razlikuje u odnosu na originalnu funkciju, moralo je da dođe do promene stanja u nekom trenutku u toku izvršavanja.
- Da li ubijanje slabom mutacijom garantuje i ubijanje jakom mutacijom?



ZADATAK 4 – TROŠKOVI DOSTAVE

REŠENJE – TEST PRIMERI

- Potrebno je pronaći test primer koji bi ubio mutanta M_2 u slučaju jake mutacije.
  - Kako je razlika u naredbi koja se izvršava ukoliko je uslov na liniji 2 zadovoljen, potrebno je da on bude zadovoljen, odnosno da težina bude veća od 10.
  - Dodatno, ne sme da se izvrši nova dodela vrednosti na liniji 5, tako da uslov na liniji 4 ne treba da bude zadovoljen.
  - Dakle, test primer koji bi ubio mutanta M_2 je:

    (weight = 12, express = False)

```
def shipping_cost(weight, express):
1.   cost = 5
2.   if weight > 10:
3.      cost += 5
4.   if express:
5.      cost = 12
6.   return cost
```

Mutant M_2

UNIVERZITET U BEOGRADU
ELEKTROTEHNIČKI
FAKULTET

PITANJA?

https://si3ts.etf.bg.ac.rs/

nikolas@etf.bg.ac.rs

