Vežba 1 - Primer GIT

Zadatak 1. Kreirati javni repozitorijum na Github-u pod nazivom *RSZDMK*. Nakon toga, potrebno je klonirati repozitorijum na Vaš računar. U repozitorijumu je potrebno napraviti folder *Vezba1*, zatim folder *Zadatak1* u *Vezba1*. U okviru *Zadatak1* foldera potrebno je napraviti datoteku *main.c*, čiji je sadržaj dat u nastavku:

```
#include <stdio.h>

int main()
{
    printf("Hello World");

    return 0;
}
```

Nakon izmene sadržaja ove datoteke, potrebno ju je postaviti u repozitorijum na gitu kao "Initial commit". Potom je potrebno izmeniti sadržaj ove datoteke sa sledećim kodom:

```
#include <stdio.h>

int main()
{
     int a = 5;
     int b = 4;
     int c;

     if(a > b)
          c = a - b;
     else
          c = b - a;

     printf("Result: %d\n", c);

    return 0;
}
```

Izmenu je potrebno postaviti na git pod nazivom "Initial code added". Sledeći korak podrazumeva izmenu vrednosti promenljivih *a* i *b* u 3 i 8 respektivno. Nakon izmene potrebno je komandom *git diff main.c* proveriti koje su izmene realizovane u odnosu na prethodnu objavu. Izmene postaviti na git pod nazivom "a and b values changed".
Naredni korak podrazumeva kreiranje nove grane pod nazivom *branch1*. Izmene koje je potrebno izvršiti na ovoj grani se sastoje od kreiranja nove funkcije pod nazivom *Calculation* koja je prikazana u sledećem kodnom segmentu i izmene sadržaja *main* funkcije tako da se umesto ovog dela koda poziva kreirana funkcija.

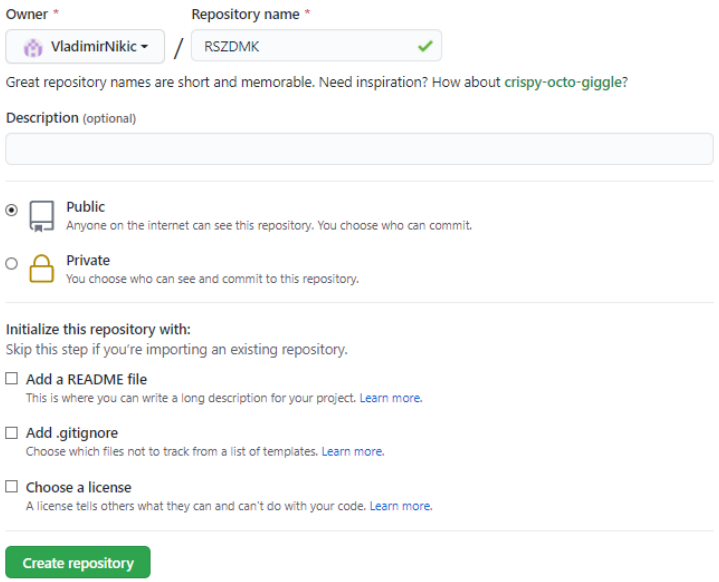
```
int Calculation(int a, int b)
{
	if(a > b)
		return a - b;
	else
		return b - a;
}
```

Izmene postaviti na grani *branch1* pod nazivom "Calculation function added". Na grani *master* je potrebno kreirati u okviru datoteke *main.c* funkciju *PrintResult* prikazanu u sledećem kodnom segmentu i zameniti postojeći ispis u *main*-u datom funkcijom. Izmene postaviti pod nazivom "PrintResult function added".

```
void PrintResult(int res)
{
	printf("Result: %d\n", res);
}
```

Sledeći korak podrazumeva spajanje *master* i *branch1* grane gde je potrebno rešiti konflikte tako što se izmene iz obe datoteke uvrste u konačnu verziju. Novu verziju postaviti pod imenom "master and branch1 merged".
Nakon toga potrebno je kreirati datoteku *end.txt* i izbrisati datoteku *main.c* iz repozitorijuma i postaviti novokreiranu datoteku *end.txt* pod nazivom "main.c removed, end.txt added".

Rešenje:
Kreirati nalog na sajtu https://github.com i prijaviti se. Klikom na dugme *New* otvara se sledeći meni. Potrebno je uneti ime repozitorijuma koji želimo da kreiramo, odabrati da li će repozitorijum biti dostupan svima (*Public*) ili samo odabranim korisnicima (*Private*) i po potrebi uneti opis onoga šta će biti smešteno u njemu.



Korak 1. Kreiranje repozitorijuma na GitHub-u

Klikom na dugme *Create repository* će repozitorijum biti napravljen, nakon čega će biti prikazan link do tog repozitorijuma. Ovaj link služi za kreiranje lokalnih kopija repozitorijuma i njihovim povezivanjem sa originalom koji se nalazi na serveru. Kreiranje lokalne kopije je prikazno u koraku 2.

```
$ git clone "https://github.com/VladimirNikic/RSZDMK.git"
Cloning into 'RSZDMK'...
warning: You appear to have cloned an empty repository.
```

Korak 2. Kloniranje repozitorijuma

Unutar lokalnog repozitorijuma napravljen je folder *Vezba1*, a zatim unutar njega folder *Zadatak1* i u njemu datoteka *main.c*. Unosimo sadržaj iz postavke zadatka u datoteku i sačuvavamo je. Upotrebom komande *git status* proveravamo stanje datoteka u repozitorijumu gde vidimo da se folder *Vezba1* i njegov sadržaj ne prati.

```
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Vezba1/

nothing added to commit but untracked files present (use "git add" to track)
```

Korak 3. Provera statusa

Kako bismo krenuli da pratimo ovaj folder i njegov sadržaj, potrebno je da ga prebacimo u pripremnu zonu koristeći komandu *git add*. Sledeća komanda nam omogućava prebacivanje samo fajla *main.c* i niza foldera u kojima se on nalazi, bez ostalog sadržaja koji se nalazi u njima.

```
$ git add Vezba1/Zadatak1/main.c
```

Korak 4. Prebacivanje izmenjenih datoteka iz repozitorijuma u pripremnu zonu

Sačuvavanje izmena repozitorijuma izvršava se upotrebom komande *git commit* koja kao parametar prima poruku. U ovom slučaju koristi se poruka definisana u postavci zadatka.

```
$ git commit -m "Initial commit"
[master (root-commit) 115967b] Initial commit
 1 file changed, 8 insertions(+)
 create mode 100644 Vezba1/Zadatak1/main.c
```

Korak 5. Sačuvavanje promena

Nakon ove komande izmene su sačuvane na lokalnom računaru, dok se izmene na originalnom repozitorijumu koji se nalazi na serveru. Ažuriranje repozitorijuma na serveru izvršava se upotrebom komande *git push origin master* gde se ažurira *master* grana repozitorijuma.

```
$ git push origin master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (5/5), 350 bytes | 350.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/VladimirNikic/RSZDMK.git
 * [new branch]      master -> master
```

Korak 6. Ažuriranje repozitorijuma

Nakon objavljivanja *Hello world!* sadržaja direktorijuma, potrebno ga je zameniti sadržajem datim u postavci. Upotrebom komande *git status* vidimo da je promena datoteke *main.c* primećena.

```
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Vezba1/Zadatak1/main.c

no changes added to commit (use "git add" and/or "git commit -a")
```

Korak 7. Provera statusa

Korišćenjem komande *git add* * prebacujemo sve foldere i datoteke, koje se nalaze unutar foldera u kom se nalazimo, u pripremnu zonu. Karakter * (*Asterisk*) je specijalni karakter koji omogućava ovu funkcionalnost.

```
$ git add *
```

Korak 8. Prebacivanje izmenjenih datoteka iz repozitorijuma u pripremnu zonu

Nakon toga sačuvavamo promene uz odgovarajuću poruku koja opisuje koje promene su izvršene u odnosu na prethodno sačuvavanje.

```
$ git commit -m "Initial code added"
[master 114ede7] Initial code added
 1 file changed, 11 insertions(+), 2 deletions(-)
```

Korak 9. Sačuvavanje promena

Repozitorijum na serveru se zatim ažurira sačuvanim izmenama.

```
$ git push origin master
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (5/5), 429 bytes | 429.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/VladimirNikic/RSZDMK.git
   115967b..114ede7  master -> master
```

Korak 10. Ažuriranje repozitorijuma

Posle izmena promenljivih *a* i *b* u 3 i 8 respektivno, promene u odnosu na prethodno sačuvane datoteke u repozitorijumu možemo videti upotrebom komande *git diff*. Kao parametar prosleđujemo ime datoteke čiji sadržaj želimo da uporedimo. Crvenom bojom je označen sadržaj koji se nalazi u poslednjoj sačuvanoj verziji datoteke, dok se zelenom bojom označavaju izmene tog sadržaja.

```
$ git diff Vezba1/Zadatak1/main.c
diff --git a/Vezba1/Zadatak1/main.c b/Vezba1/Zadatak1/main.c
index 601ed07..6928081 100644
--- a/Vezba1/Zadatak1/main.c
+++ b/Vezba1/Zadatak1/main.c
@@ -2,8 +2,8 @@

 int main()
 {
-       int a = 5;
-       int b = 4;
+       int a = 3;
+       int b = 8;
        int c;

        if(a > b)
```

Korak 11. Pregled izmena

Izmene prebacujemo u pripremnu zonu kako bismo mogli da ih sačuvamo.

```
$ git add *
```

Korak 11. Prebacivanje izmenjenih datoteka iz repozitorijuma u pripremnu zonu

Nakon čega ih sačuvavamo uz odgovarajući deskriptivni komentar.

```
$ git commit -m "a and b values changed"
[master 60da742] a and b values changed
 1 file changed, 2 insertions(+), 2 deletions(-)
```

Korak 12. Sačuvavanje promena

Promenama ažuriramo repozitorijum na master grani na serveru.

```
$ git push origin master
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (5/5), 359 bytes | 359.00 KiB/s, done.
Total 5 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/VladimirNikic/RSZDMK.git
   114ede7..60da742  master -> master
```

Korak 13. Ažuriranje repozitorijuma

Nakon toga, shodno postavci zadatka potrebno je kreirati novu granu u okviru našeg repozitorijuma pod nazivom *branch1*. Ovo je urađeno upotrebom naredbe *git branch branch1*, gde parametar *branch1* predstavlja naziv grane koju kreiramo.

```
$ git branch branch1
```

Korak 14. Kreiranje grane *branch1*

Budući da smo u ovom trenutku pozicionirani na glavnoj (*master*) grani, potrebno je izvršiti prelaz na granu *branch1* koju smo kreirali u prethodnom koraku. Ovo se može učiniti upotrebom komande *git checkout branch1*. Nakon poziva ove komande, ispisuje se poruka koja signalizira uspešnu promenu grane.

```
$ git checkout branch1
Switched to branch 'branch1'
```

Korak 15. Prelazak na granu *branch1*

U narednom koraku, potrebno je izmeniti sadržaj *main.c* datoteke odgovarajućim kodom (funkcija *Calculation*) datim u postavci zadatka. Tako izmenjen kod, potrebno je prebaciti u pripremnu zonu kako bi izmene mogli da sačuvamo.

```
$ git add *
```

Korak 16. Prebacivanje izmenjenih datoteka iz repozitorijuma u pripremnu zonu

Na sličan način, promene sačuvavamo uz odgovarajući deskriptivni komentar u skladu sa postavkom zadataka.

```
$ git commit -m "Calculation function added"
[branch1 e3e988b] Calculation function added
 1 file changed, 10 insertions(+), 4 deletions(-)
```

Korak 17. Sačuvavanje promena

Nakon sačuvavanja izmena, potrebno je ažurirati repozitorijum na serveru, ali ovog puta na grani *branch1*.

```
$ git push origin branch1
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (5/5), 462 bytes | 462.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'branch1' on GitHub by visiting:
remote:      https://github.com/VladimirNikic/RSZDMK/pull/new/branch1
remote:
To https://github.com/VladimirNikic/RSZDMK.git
 * [new branch]      branch1 -> branch1
```

Korak 18. Ažuriranje repozitorijuma

Nakon ažuriranja *branch1* grane, ponovo je potrebno pozicionirati se na glavnu (*master*) granu upotrebom komande *git checkout master*. Potom, potrebno je izvršiti odgovarajuće promene u datoteci *main.c* (implementiranje *PrintResult* funkcije). Izmenjenu datoteku prebaciti u pripremnu zonu.

```
$ git add *
```

Korak 20. Prebacivanje izmenjenih datoteka iz repozitorijuma u pripremnu zonu

Izmene se, zatim, sačuvavaju uz odogovarajući komentar dat u tekstu zadatka.

```
$ git commit -m "PrintResult function added"
[master e837a36] PrintResult function added
 1 file changed, 7 insertions(+), 1 deletion(-)
```

Korak 21. Sačuvavanje promena

Glavna grana repozitorijuma na serveru se zatim ažurira ovim izmenama.

```
$ git push origin master
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (5/5), 465 bytes | 465.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/VladimirNikic/RSZDMK.git
   60da742..e837a36  master -> master
```

Korak 22. Ažuriranje repozitorijuma

Kako su obe grane u ovom trenutku različite, u odnosu na trenutak pre grananja, spajanje njihovog sadržaja se izvršava upotrebom *git merge* komande. Ova komanda spaja datotetke iz grane u kojoj smo trenutno pozicionirani (*master*), sa onom koju prosleđujemo kao parametar (u ovom slučaju to je grana *branch1*).

```
$ git merge branch1
Auto-merging Vezba1/Zadatak1/main.c
CONFLICT (content): Merge conflict in Vezba1/Zadatak1/main.c
Automatic merge failed; fix conflicts and then commit the result.
```

Korak 23. Spajanje grana

U toku spajanja dveju verzija datoteki *main.c* GIT nije u stanju da prepozna da li je u pitanju jedna funkcija koja se razlikuje u imenu i sadržaju ili se radi o dve zasebne funkcije. Zbog toga, on ukazuje na konflikt. Konflikt rešava programer, tako što spajanjem novonastalu datoteku menja u skladu sa svojim potrebama, što je prikazano u koraku 25.

```
#include <stdio.h>

<<<<<<< HEAD
void PrintResult(int res)
{
    print("Result: %d\n", res);
=======
int Calculation(int a, int b)
{
    if(a > b)
        return a - b;
    else
        return b - a;
>>>>>>> branch1
}


int main()
{
    int a = 3;
    int b = 8;
    int c;

    c = Calculation(a, b);

    PrintResult(c);

    return 0;
}
```

Korak 24. Konfliktna datoteka *main.c*

```c
#include <stdio.h>

void PrintResult(int res)
{
    print("Result: %d\n", res);
}

int Calculation(int a, int b)
{
    if(a > b)
        return a - b;
    else
        return b - a;
}

int main()
{
    int a = 3;
    int b = 8;
    int c;

    c = Calculation(a, b);

    PrintResult(c);

    return 0;
}
```

Korak 25. Rešavanje konflikta

Izmenjena datoteka se nakon toga prebacuje u pripremnu zonu.

```
$ git add *
```

Korak 26. Prebacivanje izmenjenih datoteka iz repozitorijuma u pripremnu zonu

Nakon toga se izmene sačuvavaju uz odgovarajući komentar.

```
$ git commit -m "main.c removed, end.txt added"
[master be11526] main.c removed, end.txt added
```

Korak 27. Sačuvavanje promena

Konačno, repozitorijum na serveru se ažurira izmenama na glavnoj grani (*master*). Posle ovog koraka glavna grana (*master*) i grana *branch1* su spojene na serveru.

```
$ git push origin master
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (5/5), 450 bytes | 450.00 KiB/s, done.
Total 5 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/VladimirNikic/RSZDMK.git
   e837a36..be11526  master -> master
```

Korak 28. Ažuriranje repozitorijuma

Nakon spajanja grana, po uslovu zadatka, potrebno je napraviti novu datoteku end.txt i izvršiti brisanje datoteke *main.c* iz repozitorijuma, upotrebom komande *git rm*.

```
$ git rm Vezba1/Zadatak1/main.c
rm 'Vezba1/Zadatak1/main.c'
```

Korak 29. Brisanje datoteke *main.c* iz repozitorijuma

Nova datoteka *end.txt* se prebacuje u pripremnu zonu, čime se ona prvi put uvršvata u repozitorijum, ali samo na lokalnom računaru.

```
$ git add Vezba1/Zadatak1/end.txt
```

Korak 30. Prebacivanje izmenjenih datoteka iz repozitorijuma u pripremnu zonu

Nakon toga se potrebno je sačuvati izmene uz odgovarajući komentar.

```
$ git commit -m "main.c removed, end.txt added"
[master 59122ae] main.c removed, end.txt added
 2 files changed, 32 deletions(-)
 create mode 100644 Vezba1/Zadatak1/end.txt
 delete mode 100644 Vezba1/Zadatak1/main.c
```

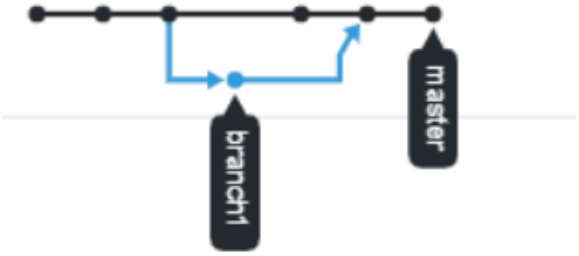
Korak 31. Sačuvavanje promena

Konačno, potrebno je izvršiti ažuriranje repozitorijuma na serveru, čime se iz ovog repozitorijuma briše datoteka *main.c*, a uvrštava datoteka *end.txt*. Datoteka *main.c* je i dalje sačuvana na prethodnim objavama repozitorijuma, međutim ona nije vidljiva u poslednjoj objavi. Upotrebom komande *git checkout* čiji je parametar broj prethodne objave, može se vratiti repozitorijum na odgovarajuću sačuvanu verziju, čime se može pristupiti različitim verzijama repozitorijuma, tj. datoteka koje se u njemu nalaze. Brojeve verzija prethodnih objava možemo dobiti upotrebom komande *git log*.

```
$ git push origin master
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Writing objects: 100% (5/5), 320 bytes | 320.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/VladimirNikic/RSZDMK.git
   be11526..59122ae  master -> master
```

Korak 32. Ažuriranje repozitorijuma

Na slici 33. prikazan je dijagram stabla koje je kreirano ovim zadatkom i predstavlja dobar vizuelni uvid u to šta je rađeno, koje objave su napravljene, koje grane su kreirane itd. Ovom dijagramu je moguće pristupiti na GitHub-u, u odeljku *Insights/Network*.



Korak 33. Prikaz dijagrama stabla koje je kreirano ovim zadatkom