Programiranje 2

Pseudo slučajni brojevi.
Programi pisani u više datoteka.

Zadatak 1 *Napraviti funkciju koja generišlučajan realan broj od 0 i 1.*

Zadatak 2 *Parametri komandne linije su* `n, a, b` *($a < b$). Treba popuniti prvih* `n` *elemenata niza* `A` *celim slučajnim brojevima koji su između* `a` *i* `b`*. Ištampati niz* `A` *na standarni izlaz. Maksimalan broj elemenata niza* `A` *je 200. Ukoliko nisu zadati svi argumenti komandne linije ili ne zadovoljavaju potrebna svojstva ispisati poruku o grešci.*

Zadatak 3 *Svaku funkciju testirati pozivom u glavnom programu.*

1. *Napisati funkciju*
   `void unos(int a[], int* n);` *(n se unosi u funkciji)*
   koja služi za unos koeficijenata polinoma. Parametar `n` *označava stepen polinoma. Prvo se unosi stepen, a potom i koeficijenti (celi brojevi). Pretpostaviti da je maksimalan stepen polinoma 100.*

2. *Koeficijenti polinoma se pamte nizu, napisati f-ju koja ispisuje polinom u obliku* `a[0] + a[1]*x + a[2]*x^2 + ... + a[n]*x^n`*. Funkcija ima prototip*
   `void ispis_polinoma(int a[], int n);`

   gde je n stepen polinoma, a ne dužina niza.

Primer 1:	***Primer 2:***
2	*3*
1 2 -3	*-5 0 0 4*
*1 + 2*x - 3*x^2*	*-5 + 4*x^3*

3. *Napisati funkciju za sumiranje dva polinoma (u opštem slučaju različitog stepena):*
   `int suma_polinoma(int a[], int n, int b[], int m, int c[]);`

   gde je `a` *niz koeficijenata prvog polinoma,* `n` *je stepen prvog polinoma,* `b` *je niz koeficijenata drugog polinoma, m je stepen drugog polinoma,* `c` *je rezultujući niz koeficijenata, i funkcija vraća veličinu niza c.*

Primer 1:
2
1 2 3
3
-5 0 0 4
*-4 + 2*x + 3*x^2 + 4*x^3*

4. *Formirati datoteke* `polinom.h, polinom.c i glavni.c`*, gde ce u polinom.h biti prototipi funkcija vezanih za polinome, u* `polinom.c` *će se "uvući" sa* `#include` *direktivom* `polinom.h` *i dati definicije ovih funkcija, a* `glavni.c` *ce biti primer "glavnog" programa koji koristi modul* `polinom.c`*.*

   Napomena:
   Kompilacija može da se radi na više načina:

   - *I način*

     ```
     gcc glavni.c polinom.c -o glavni
     ```

     Ovaj način može biti loš ako ima mnogo `.c` *fajlova, a samo jedan se promeni, posto se onda vrši ponovo kompilacija svega.*

   - *II nacin, preko* `.o` *fajlova*

     ```
     gcc -c glavni.c (proizvodi glavni.o)
     gcc -c polinom.c (proizvodi polinom.o)
     gcc glavni.o polinom.o -o glavni (linkuje glavni.o i polinom.o)
     ```

     Ovo je bolji način, pošto se samo linkuje, tj. ponovo se kompilira samo ono što je promenjeno, a linkuje se sa ostatkom, pa je skupa operacija kompilacije izbegnuta za većinu fajlova.

5. *Dodati novu funkciju u* `polinom.c` *i* `polinom.h`*, gde se polinom množi skalarom*
   `void mnoz_skalarom(int a[], int n, int c);`

Primer 1:	***Primer 2:***
2	*2*
1 2 3	*1 2 3*
-3	*0*
*-3 - 6*x - 9*x^2*	*0*

6. *Dodati novu funkciju u* `polinom.c` *i* `polinom.h`*, koja računa vrednost polinoma u tački x (koristiti Hornerovu šemu):*
   `int vr_poly(int a[], int n, int x);`

Primer 1:
2
5 2 3
3
38

7. *Dodati novu funkciju u polinom.c i polinom.h koja množi dva polinoma:*
   `int mul_poly(int a[], int n, int b[], int m, int c[])` *(funkcija vraća dimenziju niza* `c`*).*

Primer 1:
2
1 2 3
3
-5 0 2 4
*-5 - 10*x -13*x^2 + 8*x^3 + 14*x^4 + 12*x^5*

Zadatak 4 *Napisati malu biblioteku za rad sa velikim prirodnim brojevima (biblioteku razdvojiti u* `*.c` *i* `*.h` *datoteku). Sve vreme, paralelno sa razvojem funkcija, pisati i glavni program koji ih testira. Velike brojeve čitati iz datoteke čije ime se zadaje kao argument komadne linije. U svakom redu datoteke je jedan veliki broj. Upotrebiti ovu biblioteku za izračunavanje vrednosti* `100!`.

a) Definisati strukturu `VelikiBroj` *kojom se broj reprezentuje nizom cifara (najviše* `1000`*).*

b) Napisati funkciju za učitavanje velikog broja iz datoteke: `VelikiBroj ucitaj_broj(FILE* f)`

c) Napisati funkciju za ispis velikog broja u datoteku `velikibroj.txt`*:* `void(VelikiBroj b)`

Primer 1:
./a.out input.dat
input.dat:
78900876534492911000111010183736454474889499227267537

velikibroj.txt:
78900876534492911000111010183736454474889499227267537

d) Napisati funkciju za poređenje dva velika broja (funkcija vra ća -1, 0, ili 1).

Primer 1:
./a.out input.dat
input.dat:
78900876534492911000111010183736454474889499227267537
981171820201817

1

Primer 2:
./a.out input.dat
input.dat:
78900876534492911000111010183736454474889499227267537
78900876534492911000456660183736454474889499227267537

-1

e) Napisati funkciju za sabiranje dva velika broja: `VelikiBroj saberi(VelikiBroj a, VelikiBroj b)`

Primer 1:
./a.out input.dat
input.dat:
78900876534492911000111010183736454474889499227267537
981171820201817

78900876534492911000111010183736454475870671047469354

Primer 2:
./a.out input.dat
input.dat:
78900876534492911000111010183736454474889499227267537
78900876534492911000456660183736454474889499227267537

157801753068985822000567670367472908949778998454535074

f) Napisati funkciju za množenje velikog broja cifrom: `VelikiBroj mnozi_skalarom(VelikiBroj a, int x)`

Primer 1:
./a.out input.dat
input.dat:
78900876534492911000111010183736454474889499227267537
6

473405259206957466000666061102418726849336995363605222

g) Napisati funkciju za množenje dva velika broja (može se koristiti funkcija pod **f)***):* `VelikiBroj (VelikiBroj a, VelikiBroj b)`

Primer 1:
./a.out input.dat
input.dat:
78900876534492911000111010183736454474889499227267537
981171820201817

774153166448672404626386958833246351797256142884654569707567925147 29