Ispit iz Programiranja 2

Ispit traje **150** minuta

Napomene:

a) Pažljivo proučite Uputstvo pre popunjavanja Obrasca za odgovore.

b) Vrednost odgovora: tačan = **5**; netačan = **-1.25**; nevažeći (nula ili više zacrnjenih kružića) = **0.**

c) Na pitanjima se može osvojiti najviše **25** poena.

d) Zadaci nose po **20** poena.

I. ZADACI

1)U nekoj kompaniji, podaci o službenim automobilima se čuvaju u datoteci **`automobili.txt`**. Svaki red datoteke sadrži podatke o jednom automobilu – najpre registarski broj automobila (string od tačno 6 karaktera), a zatim naziv automobila koji se može sastojati od više reči. U datoteci **`putovanja.txt`** se nalaze podaci o službenim putovanjima – u jednom redu datoteke se nalazi datum (u formatu **`dd-mm-gggg`**), registarski broj korišćenog automobila i broj pređenih kilometara (ceo broj). Napisati program na programskom jeziku C koji pročita sadržaj navedenih tekst datoteka i ispiše na glavnom izlazu registarski broj i naziv automobila koji su prešli najmanji i najveći broj kilometara. U slučaju postojanja više automobila koji su prešli isti broj kilometara, uzima se u obzir onaj koji se pojavljuje prvi u datoteci.

2)Napisati program na programskom jeziku C koji u zadatom nizu znakova (stringu) ispravlja sve štamparske greške kod kojih prva dva znaka reči čine velika slova tako što drugi znak reči pretvori u malo slovo (npr. JUn treba da se zameni sa Jun). Reči su razdvojene pomoću jedne ili više razmaknica. Potrebno je realizovati funkcije za unos, obradu i ispis rezultujućeg stringa. Prilikom unosa, dužina stringa nije poznata unapred, već se čitanje vrši sve dok se ne unese znak za kraj reda. Napisati glavni program koji pozove funkcije za unos stringa, obradu i ispis rezultata. Program treba da ponavlja izvršavanje sve dok se sa glavnog ulaza ne unese prazan string. **Napomena**: komunikacija između glavnog programa i funkcija, kao i između samih funkcija treba da se obavlja isključivo preko argumenata i povratnih vrednosti. Voditi računa o ispravnoj alokaciji i dealokaciji dinamičke memorije.

II. PITANJA

1)Koja od sledećih naredbi na jeziku C ispravno definiše pokazivač na funkciju koja prihvata argument tipa niz pokazivača na podatke tipa **`double`**, a vraća podatak tipa pokazivač na **`char`**?

A)
```
char **x(double *y);
```
B)
```
double *(*a)(char b[]);
```
(C)
```
char *(*p)(double *f[]);
```

2)Šta ispisuje sledeći program na programskom jeziku C (pod pretpostavkom da dinamička alokacije memorije uvek uspeva)?

```
#include <stdio.h>
#include <stdlib.h>
void main(){
int *p = malloc(sizeof(int)), *q = malloc(sizeof(int)), *r, m, n;
    n = 1; m = 2; *p = m; *q = n, r = &n;
    for(n=1,m=3; n<5; n++) switch((*p * *r)%9 ){
                              case 0: case 2: case 3: m++;
                              case 4: case 5: case 6: m+=2; break;
                              default: m=n;
                           }
    printf("%d%d%d%d", m, n, *p, *q); free(p); free(q);
}
```

A) 5431 **(B)** 4521 C) 3223

3)Šta ispisuje sledeći program na programskom jeziku C (pod pretpostavkom da dinamička alokacije memorije uvek uspeva)?

```
#include <stdio.h>
#include <string.h>
void funkcija(int x, int b) {
   if (x) {
         funkcija(x/b,b);
         b++; x++;
   }
```

```
void main() {
   int m,n;
   m = strlen("Programiranje_2");
   n = strlen("ETF");
   funkcija(m,n);
}
```

```
   printf("%d",x%b);
}
```

(A) 0220 B) 0211 C) 1211

4) Šta radi sledeći fragment koda na programskom jeziku C, pod pretpostavkom da dinamička alokacija memorije uvek uspeva? Polja **prvi**, **posl** i **tek** strukture **Lista** pokazuju na prvi, poslednji i tekući element liste, respektivno. Smatrati da lista nije prazna i da **tek** nije **NULL**.

```
typedef struct elem { int broj; struct elem *pret, *sled; } Elem;
typedef struct { Elem *prvi, *posl, *tek; } Lista;
void dodaj (Lista *lst, int b) {
  Elem *novi = malloc (sizeof(Elem));
  novi->broj = b;
  novi->pret = lst->tek->pret;
  novi->sled = lst->tek;
  if (! lst->tek->pret) lst->prvi = novi;
  else lst->tek->pret->sled = novi;
  lst->tek = lst->tek->pret = novi;
}
```

(A) Dodaje element u dvostruko ulančanu listu ispred tekućeg elementa i dodati element postaje tekući.

B) Dodaje element u dvostruko ulančanu listu iza tekućeg elementa i dodati element postaje tekući.

C) Dodaje element u dvostruko ulančanu listu na početak liste i dodati element postaje tekući.

5) Šta ispisuje sledeći fragment koda na programskom jeziku C?

```
#include <stdio.h>
typedef int (*func)(int *);
int p1(int *a) {
   return *a++;
}
int p2(int *b) {
   return *b*2;
}
int p3(int *c) {
   return c[0]+c[1]+c[2];
}
```

```
main() {
func nf[] = { p1, p2, p3 };
func nf2[6] = { 0 };
func *pf;
int d[4][4] = { { 0, 1, 2 }, { 3, 4, 5} };
int i;
  for(i=0; i<3; i++) nf2[i<<1] = nf[i];
  for(pf=nf2+5,i=6;pf>=nf2;i--,pf--)
    if( *pf )
      printf("%d",(*pf)(&d[i%3][(i+2)%3]));
}
```

A) 042 B) 003 **(C)** 043

6) Šta ispisuje sledeći fragment koda na programskom jeziku C, ako su argumenti programa vrednosti 5 6 3 7 10?

```
#include <stdio.h>
#include <stdlib.h>
main(int argc, char *argv[]) {
int i, a, b, c = 0, d;
      for(i=1;i<argc;i++) {
            d=atoi(argv[i]);
            if(i==1) a = b = d;
            else { if( a<d ) a=d;
                   if( b>d ) b=d; }
            c += d;  }
      printf("%d%d%d", a, b, c);
}
```

A) 3106 **(B)** 10331 C) 356