ELEKTROTEHNIČKI FAKULTET UNIVERZITETA U BEOGRADU BEOGRAD, 18.09.2018.

Ispit iz Programiranja 2

Ispit traje 135 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 **20** poena. Prvi zadatak nosi **20** poena, dok drugi nosi **25** poena.

I ZADACI

1)Potrebno je napisati program na programskom jeziku C koji ispisuje jedan zatvoreni ciklus u igri dodavanja loptom. Igru započinje prvi igrač koji loptu dodaje sledećem najbližem igraču. Zatim taj igrač loptu dodaje sledećem najbližem i itd., pod uslovom da to nije isti onaj koji mu je upravo dodao loptu. Napisati potprogram **`int findTheNearestOne(int *distancesFromOthers, int playersCnt, int lastPlayerIndex)`** koji određuje indeks prvog igrača koji je najbliži tekućem, znajući razdaljine između tekućeg i svih ostalih igrača (uključujući i tekućeg, kada je razdaljina 0), ukupnog broja igrača u igri, kao i indeks igrača od kojeg je tekući primio loptu (vrednost `-1` označava da nije bilo prethodnog igrača, jer je igra tek započeta). Napisati program koji ispisuje indekse igrača po redosledu dodavanja u igri, ukoliko su informacije o razdaljinama između igrača date u fajlu **`playerDistances.txt`**. Glavni program treba najpre iz prvog reda datoteke da učita indeks igrača koji započinje igru, kao i broj igrača N, a zatim da iz narednih N redova čita po N vrednosti koje redom predstavljaju udaljenosti svakog pojedinačnog igrača od svih ostalih (uključujući i sebe, kada je razdaljina 0). Nakon učitavanja informacija o razdaljinama, program treba da odredi jedan zatvoreni ciklus igrača u igri po redosledu dodavanja loptom, čije indekse ispisuje na standarnom izlazu. Potprogrami sa glavnim programom komuniciraju isključivo putem argumenata i povratne vrednosti. Nije potrebno proveravati ispravnost ulaznih podataka.

Primer ulazne datoteke	Primer izlaza
0 4 0 3 4 3 3 0 5 6 4 5 0 5 3 6 5 0	0 1 2 0

2)Napisati program na programskom jeziku C koji pomaže brucošu Mikiju da iznajmi stan. Spisak stanova za iznajmljivanje je dat u teksutalnoj datoteci **`stanovi.txt`**. U svakom redu se nalaze identifikacioni broj stana (ceo broj), cena iznajmljivanja (realan broj) i geografska širina i dužina lokacije na kojoj se stan nalazi (realni brojevi) u Dekartovom koordinatnom sistemu. Miki želi da živi u stanu koji se nalazi u određenom radijusu u odnosu na fakultet čije se koordinate mogu smatrati centrom posmatranog koordinatnog sistema. Program najpre treba da učita realan broj koji predstavlja radijus pretrage u odnosu na lokaciju fakulteta, pročita sadržaj ulazne datoteke i u izlaznu datoteku **`stanovi_filtrirano.txt`** ispiše sve stanove koji zadovoljavaju zadati kriterijum, uređene neopadajuće po ceni po istom formatu kao kod ulazne datoteke. Voditi računa o ispravnom korišćenju dinamičke memorije.

Primer ulazne datoteke	Primer izlazne datoteke za radijus 500 metara
54 200 150.0 200.0 17 150 550.0 250.0 33 175 280.5 350.1	33 175 280.5 350.1 54 200 150.0 200.0

II PITANJA

1)Koje od ponuđenih tvrdnji su tačne za programski jezik C?
A) Funkcija **`fopen`** vraća vrednost **`EOF`** ukoliko dođe do neuspešnog otvaranja datoteke.

(B) Funkcija **`feof`** vraća vrednost različitu od `0` samo ukoliko je indikator kraja datoteke postavljen od strane neke funkcije za čitanje datoteke.

C) Čitanje binarnih datoteka se može obaviti funkcijom **`fscanf`** korišćenjem konverzije **`%b`**.

2)Šta ispisuje sledeći program napisan na programskom jeziku C?

```
#include <stdio.h>
#include <stdlib.h>
#define N 4
void main() {
  int i, b[N] = {2, 0, 1, 8}, **a = (int**)calloc(N, sizeof(int*));
  for (i = N-1; i >= 0; i--) *(a+i) = b + N - i - 1;
  for (i = 0; i < N; i++) printf("%d ", *a[i]);
  free(a);
}
```

(A) 8 1 0 2 B) 3 1 2 9 C) 2 0 1 8

3)Šta ispisuje sledeći program napisan na programskom jeziku C ako funkcija **`free_struct`** ispravno briše svu alociranu memoriju?

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct exam {
  char text1[10];
  char *text2;
};
int main()
{
    struct exam s1, s2;
    strcpy(s1.text1, "okt");
    s1.text2 = malloc(10);
    strcpy(s1.text2, s1.text1);
   s2 = s1;
    s1.text1[2] = '1';
    s1.text2[2] = '1';

    printf("%c%c", s2.text1[2], s2.text2[2]);

    free_struct(&s1, &s2);
    return 0;
}
```

A) tt B) 1t **(C)** t1

4)Šta ispisuje sledeći program na programskom jeziku C ukoliko su u jednostruko ulančanu listu **`lst1`** redom uneseni brojevi 8 3 6 9 1, a u jednostruko ulančanu listu **`lst2`** redom uneseni brojevi 3 12 1 7? Smatrati da funkcija **`citaj`** ispravno formira, a funkcija **`pisi`** ispravno ispisuje sadržaj zadate liste redom od početka.

```
#include <stdio.h>
#include <stdlib.h>
typedef struct elem {
    int broj; struct elem *sled;
} Elem;
Elem *okt_fun(Elem *lst1, Elem *lst2);
void main () {
  Elem *lst, *lst1, *lst2;
  lst1 = citaj();
  lst2 = citaj();
  lst = okt_fun(lst1, lst2);
  pisi(lst);
  brisi(lst);
}
```

```
Elem *okt_fun(Elem *lst1, Elem *lst2) {
  Elem *lst = NULL, *posl = NULL, *tek;
  while (lst1 && lst2) {
    if (lst1->broj > lst2->broj) tek = lst1;
    else tek = lst2;
    lst1 = lst1->sled;
    lst2 = lst2->sled;
    tek->sled = NULL;
    if (!lst) lst = tek;
    else posl->sled = tek;
    posl = tek;
  }
  return lst;
}
```

(A) 8 12 6 9 B) 3 3 1 7 1 C) 8 3 12 9 7

5)Šta ispisuje sledeći program napisan na programskom jeziku C ako se sa standardnog ulaza unese **`easyguess`**?

```
#include <stdio.h>
#include <string.h>

void manipulateString(char *str) {
  char cnt[127] = {0}, *p = str, *q;
  while(*p) cnt[*p++]++;

  q = str + strlen(str) - 1;
  while(str<q) {
    while(str<q && !(cnt[*str]&1)) str++;
    while(str<q && !(cnt[*q]&1)) q--;
    if(str<q) {
      char t = *str; *str = *q; *q = t;
    }
    str++; q--;
  } //end_while
} //_end_function
int main(void) {
    char str[20];
    scanf("%s", str);
    manipulateString(str);
    printf("%s\n", str);
}
```

A) eseugysas **(B)** essugyesa C) sseugysae

13S111P2, Programiranje 2, oktobar 2017/2018.
Rešenje zadatka

Zadatak 1

Potrebno je napisati program na programskom jeziku C koji ispisuje jedan zatvoreni ciklus u igri dodavanja loptom. Igru započinje prvi igrač koji loptu dodaje sledećem najbližem igraču. Zatim taj igrač loptu dodaje sledećem najbližem i itd., pod uslovom da to nije isti onaj koji mu je upravo dodao loptu. Napisati potprogram **`int findTheNearestOne(int *distancesFromOthers, int playersCnt, int lastPlayerIndex)`** koji određuje indeks prvog igrača koji je najbliži tekućem, znajući razdaljine između tekućeg i svih ostalih igrača (uključujući i tekućeg, kada je razdaljina 0), ukupnog broja igrača u igri, kao i indeks igrača od kojeg je tekući primio loptu (vrednost **`-1`** označava da nije bilo prethodnog igrača, jer je igra tek započeta). Napisati program koji ispisuje indekse igrača po redosledu dodavanja u igri, ukoliko su informacije o razdaljinama između igrača date u fajlu **`playerDistances.txt`**. Glavni program treba najpre iz prvog reda datoteke da učita indeks igrača koji započinje igru, kao i broj igrača N, a zatim da iz narednih N redova čita po N vrednosti koje redom predstavljaju udaljenosti svakog pojedinačnog igrača od svih ostalih (uključujući i sebe, kada je razdaljina 0). Nakon učitavanja informacija o razdaljinama, program treba da odredi jedan zatvoreni ciklus igrača u igri po redosledu dodavanja loptom, čije indekse ispisuje na standarnom izlazu. Potprogrami sa glavnim programom komuniciraju isključivo putem argumenata i povratne vrednosti. Nije potrebno proveravati ispravnost ulaznih podataka.

```c
#include <stdio.h>
#include <stdlib.h>

#define FILE_ERROR 1
#define ALLOCATION_ERROR 2
#define INVALID_INDEX -1

#define COPEN_FILE(fhandle, filename, mode)\
    if((fhandle = fopen(filename, mode)) == 0) exit(FILE_ERROR);

#define CMALLOC(ptr, size)\
    if((ptr = malloc(size)) == 0) exit(ALLOCATION_ERROR);

void releaseMemory(int **a, int dim){
    for(int i = 0; i < dim; i++) free(a[i]);
    free(a);
}

void readData(FILE *fhandle, int ***distances, int *playersCnt,
                                               int *startPlayer){
    int** d;
    fscanf(fhandle, "%d %d", startPlayer, playersCnt);
    CMALLOC(d, sizeof(int*) * (*playersCnt));
    for(int i = 0; i < *playersCnt; i++){
        d[i] = malloc(sizeof(int) * (*playersCnt));
        for(int j = 0; j < *playersCnt; j++) fscanf(fhandle, "%d", &d[i][j]);
    }
    (*distances) = d;
}

int findTheNearestOne(int *distancesFromOthers, int playersCnt,
                                                int lastPlayerIndex){
    int *d = distancesFromOthers;
    int minInd = INVALID_INDEX;
    for(int i = 0; i < playersCnt; i++)
        if(d[i] != 0 && i != lastPlayerIndex &&
          (minInd == INVALID_INDEX || d[minInd] > d[i]))
            minInd = i;
    return minInd;
}
```

```c
void findLoop(int **distances, int playersCnt, int startPlayer){
    int currPlayer = startPlayer, prevPlayer = -1, nextPlayer;
    do {
        printf("%d ", currPlayer);
        nextPlayer = findTheNearestOne(distances[currPlayer],
                                       playersCnt, prevPlayer);
            prevPlayer = currPlayer;
            currPlayer = nextPlayer;
    } while (currPlayer != startPlayer);
    printf("%d", currPlayer);
}

int main(){
    FILE *distancesFile;
    int **distances;
    int playersCnt, startPlayer;

    // -- CodeLite::'playerDistances.txt' and src file are in the same dir //
    COPEN_FILE(distancesFile, "..\\playerDistances.txt", "r");
    readData(distancesFile, &distances, &playersCnt, &startPlayer);
    findLoop(distances, playersCnt, startPlayer);

    fclose(distancesFile);
    releaseMemory(distances, playersCnt);
    return 0;
}
```

Suma:	**20**

13S111P2, Programiranje 2, oktobar 2017/2018.
Rešenje zadatka

Zadatak 2

Napisati program na programskom jeziku C koji pomaže brucošu Mikiju da iznajmi stan. Spisak stanova za iznajmljivanje je dat u teksutalnoj datoteci stanovi.txt. U svakom redu se nalaze identifikacioni broj stana (ceo broj), cena iznajmljivanja (realan broj) i geografska širina i dužina lokacije na kojoj se stan nalazi (realni brojevi) u Dekartovom koordinatnom sistemu. Miki želi da živi u stanu koji se nalazi u određenom radijusu u odnosu na fakultet čije se koordinate mogu smatrati centrom posmatranog koordinatnog sistema. Program najpre treba da učita realan broj koji predstavlja radijus pretrage u odnosu na lokaciju fakulteta, pročita sadržaj ulazne datoteke i u izlaznu datoteku stanovi_filtrirano.txt ispiše sve stanove koji zadovoljavaju zadati kriterijum, uređene neopadajuće po ceni po istom formatu kao kod ulazne datoteke. Voditi računa o ispravnom korišćenju dinamičke memorije.

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

typedef struct stan {
  int id_stana;
  double cena, geo_x, geo_y;
} Stan;

typedef struct elem {
  Stan stan;
  struct elem *sled;
} Elem;
Elem *citaj_stanove(FILE *ulaz) {
  Elem *prvi, *posl, *novi;
  int id_stana;
  double cena, geo_x, geo_y, r;

  prvi = posl = NULL;
  while ((fscanf(ulaz, "%d %lf %lf %lf", &id_stana, &cena, &geo_x, &geo_y)) > 0) {
    novi = malloc(sizeof(Elem));
    if (!novi) {
      printf("Greska pri radu sa din. memorijom!\n");
      return 2;
    }
    novi->stan.id_stana = id_stana;
    novi->stan.cena = cena;
    novi->stan.geo_x = geo_x;
    novi->stan.geo_y = geo_y;
    novi->sled = NULL;
    if (!prvi) prvi = novi;
    else posl->sled = novi;
    posl = novi;
  }

  return prvi;
}
void pisi_stanove(Elem *prvi, double r, FILE *izlaz) {
  Elem *tek;
  tek = prvi;
  while (tek) {
    if (sqrt(tek->stan.geo_x*tek->stan.geo_x + tek->stan.geo_y*tek->stan.geo_y) < r)
      fprintf(izlaz, "%d %.2lf %.3lf %.3lf\n", tek->stan.id_stana, tek->stan.cena, tek-
>stan.geo_x, tek->stan.geo_y);
    tek = tek->sled;
  }
}
void sortiraj_stanove(Elem *prvi) {
  Elem *tek1, *tek2;
  tek1 = prvi;
  while (tek1->sled) {
    tek2 = tek1->sled;
    while (tek2) {
      if (tek1->stan.cena > tek2->stan.cena) {
        Stan temp = tek1->stan;
        tek1->stan = tek2->stan;
```

```
        tek2->stan = temp;
      }
      tek2 = tek2->sled;
    }
    tek1 = tek1->sled;
  }
}
void dealociraj(Elem *prvi) {
  Elem *stari;
  while (prvi) {
    stari = prvi;
    prvi = prvi->sled;
    free(stari);
  }
}
int main() {
  FILE *ulaz, *izlaz;
  Elem *prvi;
  double r;

  ulaz = fopen("stanovi.txt", "r");
  izlaz = fopen("stanovi_filtrirano.txt", "w");
  if (!ulaz || !izlaz) {
    printf("Greska pri radu sa datotekom!\n");
    return 1;
  }

  printf("Unesite radijus stana za pretragu: ");
  scanf("%lf", &r);

  prvi = citaj_stanove(ulaz);
  sortiraj_stanove(prvi);
  pisi_stanove(prvi, r, izlaz);
  dealociraj(prvi);

  fclose(ulaz);
  fclose(izlaz);

  return 0;
}
```

Suma:	**25**