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 u nizu celih brojeva dužine N nalazi K najvećih elemenata niza, gde je N > K i smešta ih u dinamički niz dužine K. Svi elementi niza imaju različite vrednosti. Potrebno je da program najpre učita brojeve N i K, a potom i same elemente niza. Nakon toga program treba da nađe i ispiše K najvećih elemenata u nizu u nerastućem poretku. Učitavanje podataka, pronalaženje K najvećih elemenata i njihov ispis izdvojiti u zasebne potprograme koji sa glavnim programom komuniciraju samo putem argumenata i povratne vrednosti. Izmena originalnog niza nije dozvoljena.

Primer ulazne datoteke	Primer izlazne datoteke
6 3 5 3 -2 4 -1 10	10 5 4

2)Napisati program na programskom jeziku C koji vrši obradu podataka o posećenosti fudbalskih utakmica. Jedan red ulazne datoteke **`tekme.txt`** sadrži podatke u sledećem formatu: domaći tim (jedna reč od najviše 20 slova), gostujući tim (jedna reč od najviše 20 slova) i poseta na utakmici (ceo broj). Podaci su odvojeni po jednim blanko znakom. Napisati program koji za svaki tim određuje prosečnu posetu kada je taj tim bio domaćin i kada je taj tim bio gost. Za tim koji nikad nije bio domaćin, odnosno gost, prosečna poseta za tu ulogu treba da bude 0. Podaci se ispisuju u izlaznu datoteku **`poseta.txt`** razdvojeni jednim blanko znakom u sledećem formatu: tim, prosečna poseta kada je taj tim bio domaćin i prosečna poseta kada je taj tim bio gost. Voditi računa o ispravnom korišćenju dinamičke memorije.

Primer ulazne datoteke **`tekme.txt`**	Primer izlazne datoteke **`poseta.txt`**
Zvezda Partizan 20000 Zvezda Rad 15000 YoungBoys Zvezda 26000 Partizan Rad 10000	Zvezda 17500 26000 Partizan 10000 20000 Rad 0 12500 YoungBoys 26000 0

II PITANJA

1)Šta je sadržaj jednostruko ulančane liste koju vrati funkcija **`run`** napisana na programskom jeziku C ukoliko joj se putem parametra **`list`** prosledi lista koja sadrži redom brojeve 5 4 2 4 2, a za parametar **`size`** 5?

```
#include <stdio.h>
#include <stdlib.h>
#define swap(a,b) do {int t = (a); \
        (a) = (b); (b) = t;} while(0)
typedef struct elem {
   int data;
   struct elem *next;
} Elem;
void combine(Elem *list1, Elem *list2) {
   Elem *curr = list1;
   for (; curr->next; curr = curr->next);
   curr->next = list2;
}
Elem* run(Elem *list, int size) {
   if (size < 2) {
      return list;
   } else if (size < 3) {
      if (list->data > list->next->data) {
         swap(list->data, list->next->data);
      }
      return list;
   } else {
      Elem *curr = list, *tmp = list;
      int rs = 1;
      while(curr) {
         curr = curr->next;
         if (curr) {
            curr = curr->next;
            tmp = tmp->next;
            rs++;
         }
      }
      curr = tmp->next;
      tmp->next = 0;
      list = run(list, rs);
      curr = run(curr, size - rs);
      if (list->data > curr->data) {
         combine(curr, list);
         return curr;
      }
      combine(list, curr);
      return list;
   }
}
```

A)2 2 4 4 5 **(B)** 2 4 5 2 4 C) 4 5 2 2 4

2)Neka se posmatra funkcija **my_strcat** napisana na programskom jeziku C koja ispravno nadovezuje string **src** na kraj stringa **dst**. Koja tvrđenja su tačna?

```
char* my_strcat(char* dst, const char* src) {
  char* ptr = dst + strlen(dst);
  while (*src) *ptr++ = *src++;
  *ptr = '\0';
  return dst;
}
```

A) Izraz **char* ptr = dst + strlen(dst)** postavlja pokazivač **ptr** na jednu poziciju iza terminatora stringa **dst** sa vrednošću nula.

(B) Uslov **while** petlje ***src** se može zameniti sa ***src != '\0'** uz zadržavanje korektnosti funkcije.

C) Izraz ***ptr++ = *src++** se može zameniti sa ***++ptr = *++src** uz zadržavanje korektnosti funkcije.

3)Šta ispisuje sledeći program na programskom jeziku C?

```
#include <stdio.h>
int f0(int x) {return (x*9+1)%5;}
int f1(int x) {return (x<<2)%5;}
int f2(int x) {return (x*8)%5;}
```

```
void main(){
    int s = 3, i = 4, j;
    int (*f[5])(int) = {f1, f2, f0, f2, f0};
    for (j = 0; j < 5; j++) s += i = (*f[i])(i);
    printf("%d", s);
}
```

A) 15 **(B)** 17 C) 19

4)Šta ispisuje sledeći program napisan na programskom jeziku C ukoliko se na standardnom ulazu prosledi 4 4? Funkcija **dealloc** oslobađa dinamički zauzeti prostor.

```
#include <stdio.h>
#include <stdlib.h>
int** alloc(int *m, int *n) {
  scanf("%d%d", m, n);
  int **mat = malloc(*m * sizeof(*mat));
  for (int i = 0; i < *m; i++) {
    mat[i] = malloc(*n * sizeof(**mat));
    for (int j = 0; j < *n; j++)
      mat[i][j] = (i * j + 1) % *m;
  }
  return mat;
}
void dealloc(int **mat, int m);
void foo(int **mat, int m, int n) {
  if(n < 3) return;
  int ms = -1, cnt = n - 2;
  int **end = mat + m - 2;
  for(; mat < end; mat++) {
    int *r = *mat, *p = *(mat + 1) + 1,
        *q = *(mat + 2);
    int i = cnt, s = 0;
    s += *r + *(r + 1) + *q + *(q + 1);
    do {
      s += *(r + 2) + *p + *(q + 2);
      if(s > ms) ms = s;
      s -= *r++ + *p++ + *q++;
    } while(--i > 0);
  }
  printf("%d", ms);
}
int main(void) {
  int m, n, **mat = alloc(&m, &n);
  foo(mat, m, n); dealloc(mat, m);
}
```

(A) 13 B) 9 C) 11

5)Šta ispisuje sledeći program napisan na programskom jeziku C ukoliko se pozove komandom prog 6 4 2 3 2 1 1?

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[]) {
  int *arr, i;
  if (argc < 2) return 1;
  arr = calloc(atoi(argv[1]), sizeof(int));
  for (i = argc - 1; i > 1; i--)
    arr[atoi(argv[i])]++;
  for (i = 0; i < atoi(argv[1]); i++)
    printf("%d", arr[i]);
  return 0; }
```

A) 002211 B) 221100 **(C)** 022110

13E111P2, Programiranje 2, avgust 2018/2019.
Rešenje zadatka

Zadatak 1.

Potrebno je napisati program na programskom jeziku C koji u nizu celih brojeva dužine N nalazi K najvećih elemenata niza, gde je N > K i smešta ih u dinamički niz dužine K. Svi elementi niza imaju različite vrednosti. Potrebno je da program najpre učita brojeve N i K, a potom i same elemente niza. Nakon toga program treba da nađe i ispiše K najvećih elemenata u nizu u nerastućem poretku. Učitavanje podataka, pronalaženje K najvećih elemenata i njihov ispis izdvojiti u zasebne potprograme koji sa glavnim programom komuniciraju samo putem argumenata i povratne vrednosti. Izmena originalnog niza nije dozvoljena.

Primer ulazne datoteke	Primer izlazne datoteke
6 3 5 3 -2 4 -1 10	10 5 4

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

void readArray ( int *array, int n ) {
    for ( int i = 0; i < n; ++i ) {
        scanf ( "%d", &array[i] );
    }
}
int* findKMax ( int *array, int n, int k ) {
    int *kMax       = calloc ( k, sizeof ( int ) );
    int found       = 0;
    int previousMax = 0;

    if ( kMax == NULL ) {
        exit ( 1 );
    }

    for ( int i = 0; i < k; ++i ) {
        int max = 0, first = 1;

        for ( int j = 0; j < n; ++j ) {
            if ( first == 1 &&  ( array[j] < previousMax || found == 0 ) ) {
                max = array[j];
                first = 0;
            } else if ( array[j] > max
                        && ( array[j] < previousMax || found == 0 ) ) {
                max = array[j];
            }
        }

        previousMax = max;
        kMax[found] = max;
        found++;
    }

    return kMax;
}
void printArray ( int *array, int n ) {
    for ( int i = 0; i < n; ++i ) {
        printf ( "%d ", array[i] );
    }
    printf ( "\n" );
}
```

```c
int main ( ) {
    int n, k;
    scanf ( "%d %d", &n, &k );

    int *array = calloc ( n, sizeof ( int ) );
    if ( array == NULL ) {
        exit ( 1 );
    }

    readArray ( array, n );
    int *kMax = findKMax ( array, n, k );
    printArray ( kMax, k );

    free ( array );
    free ( kMax );
    return 0;
}
```

13E111P2, Programiranje 2, septembar 2018/2019.
Rešenje zadatka

Zadatak 2.

Napisati program na programskom jeziku C koji vrši obradu podataka o posećenosti fudbalskih utakmica. Jedan red ulazne datoteke **tekme.txt** sadrži podatke u sledećem formatu: domaći tim (jedna reč od najviše 20 slova), gostujući tim (jedna reč od najviše 20 slova) i poseta na utakmici (ceo broj). Podaci su odvojeni po jednim blanko znakom. Napisati program koji za svaki tim određuje prosečnu posetu kada je taj tim bio domaćin i kada je taj tim bio gost. Za tim koji nikad nije bio domaćin, odnosno gost, prosečna poseta za tu ulogu treba da bude 0. Podaci se ispisuju u izlaznu datoteku **poseta.txt** razdvojeni jednim blanko znakom u sledećem formatu: tim, prosečna poseta kada je taj tim bio domaćin i prosečna poseta kada je taj tim bio gost. Voditi računa o ispravnom korišćenju dinamičke memorije.

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

#define NAME_LEN 21
#define HOME 0
#define AWAY 1
#define SIDES 2

typedef struct team {
    char name[NAME_LEN];
    int att[SIDES], cnt[SIDES];
} Team;

typedef struct node {
    Team team;
    struct node *next;
} Node;
Node* findTeam(Node *list, char *name) {
    while(list && strcmp(list->team.name, name))
        list = list->next;

    return list;
}

Node* loadData() {
    Node *first = NULL, *last = NULL;
    FILE *file = fopen("tekme.txt", "r");
    if(!file) {
        perror(NULL); exit(1);
    }

    char teams[2][NAME_LEN];
    int att;
    while(fscanf(file, "%s %s %d", teams[HOME], teams[AWAY], &att) == 3) {
        for(int i = 0; i < SIDES; i++) {
            Node *temp = findTeam(first, teams[i]);
            if(!temp) {
                temp = calloc(sizeof(Node), 1);
                if(!temp) {
                    perror(NULL); exit(1);
                }
                strcpy(temp->team.name, teams[i]);
                if(!first) first = temp;
                else last->next = temp;
                last = temp;
            }
            temp->team.att[i] += att;
            temp->team.cnt[i]++;
        }
    }

    fclose(file);
    return first;
}
```

```
void printData(Node *list) {
    FILE *file = fopen("poseta.txt", "w");
    if(!file) {
        perror(NULL);
        exit(1);
    }

    double home_avg, away_avg;
    while(list) {
        home_avg = list->team.cnt[HOME] ?
                   (double)list->team.att[HOME] / list->team.cnt[HOME] : 0;
        away_avg = list->team.cnt[AWAY] ?
                   (double)list->team.att[AWAY] / list->team.cnt[AWAY] : 0;
        fprintf(file, "%s %.2f %.2f\n", list->team.name , home_avg, away_avg);
        list = list->next;
    }

    fclose(file);
}
void freeData(Node *first) {
    Node *old;
    while(first) {
        old = first;
        first = first->next;
        free(old);
    }
}

int main(void) {
    Node *first = loadData();
    printData(first);
    freeData(first);
}
```