Ispit iz Programiranja 2

Šifra zadatka

60 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.

PITANJA

Pitanje

Na nekom računaru realni brojevi se predstavljaju na 9 bita u formatu **seeeemmmm**, gde je **s** bit za predznak broja, **eeee** biti eksponenta u kodu sa viškom 7, a **mmmm** biti normalizovane mantise sa skrivenim bitom ($1 \leq M < 2$). Celi brojevi se predstavljaju na širini od 8 bitova u komplementu dvojke. Izgled celobrojne promenljive, smeštene u lokaciju A, je $3D_{16}$, a izgled realne promenljive, smeštene u lokaciju B, je 625_8. Ukoliko se izvrši operacija C = A - B, kako će izgledati predstava rezultujućeg realnog broja u lokaciji C nakon izračunavanja te operacije? Sva zaokruživanja obavljati prema pravilima ANSI/IEEE standarda za realne brojeve.

Ⓡ) $1D8_{16}$ (Ⓡ) 320_8 Ⓡ) 011000001_2

Pitanje

Šta ispisuje sledeći program napisan na programskom jeziku C?

```
#include <stdio.h>
void main() {
   unsigned int i=0x17, cnt = 0;
   while(i & (i - 1)) {
      switch (i & 03) {
         case 0: i >>= 1; break;
         case 1: i | (i + 1);
         case 2: i ^= 1; continue;
         default: i += 1;
      }
      cnt++;
   }
   printf("%d", cnt);
}
```

(Ⓡ) 4 Ⓡ) 3 Ⓡ) 5

Pitanje

Šta ispisuje sledeći program napisan na programskom jeziku C ako se putem standardnog ulaza prosledi string **sansa**?

```
#include <stdio.h>
#include <string.h>
#define MAX 10
void fun(char *str, char *q) {
   char *p = str, *r = q - 1, t;
   if(q - p <= 0) return;
   while(1) {
      while(*p < *q) p++;
      while(r > p && *r >= *q) r--;
      if(p < r) { t = *p; *p = *r; *r = t; }
      else break;
   }
   t = *q; *q = *p; *p = t;
   fun(str, p - 1);
   fun(p + 1, q);
}
int main(void) {
   char s[MAX];
   scanf("%s", s);
   fun(s, s + strlen(s) - 1);
   printf("%s", s);
}
```

(Ⓡ) aanss Ⓡ) saasn Ⓡ) asnas

Pitanje

Šta ispisuje sledeći program napisan na programskom jeziku C? Program je pokrenut sledećom komandom: **./p2.exe junski ispitni rok 2020**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main(int argc, char *argv[]) {
   int len = argc - 1, left = 0, right = len;
   char **str = malloc(len * sizeof(*str));
   for (int i = 1, j; i < argc; i++) {
      if(strlen(argv[0]) > strlen(argv[i])) {
         str[--right] = argv[i];
         continue;
      }
      for(j = left - 1; j >= 0; j--) {
         if(strcmp(str[j], argv[i]) <= 0)
            break;
         str[j + 1] = str[j];
      }
      str[j + 1] = argv[i]; left++;
   }
   for (int i = 0; i < len; i++)
      printf("%s ", str[i]);
   free(str);
}
```

(®) 2020 rok ispitni junski ®) rok 2020 junski ispitni ®) junski ispitni 2020 rok

Pitanje

Šta radi sledeća funkcija napisan na programskom jeziku C nad jednostruko ulančanom listom celih brojeva? Pretpostaviti da lista nije prazna.

```
typedef struct elem {
    int broj;
    struct elem *sled;
} Elem;
void listFunc(Elem *lst) {
  Elem *i;
  for (i = lst; i->sled != NULL; i = i->sled)
    if (i->broj > i->sled->broj) {
      int tmp = i->broj;
      i->broj = i->sled->broj;
      i->sled->broj = tmp;
    }
```

®) Obrće redosled elemenata u listi.
®) Uređuje zadatu listu u nerastućem poretku.
(®) Premešta na kraj liste najveći element pri čemu se ne garantuje očuvanje redosleda ostalih elemenata.

Ispit iz Programiranja 2

II Deo

90 minuta

Napomene:
a) Prvi zadatak nosi **20** poena, dok drugi nosi **25** poena.

ZADACI

Zadatak 1

Napisati program na programskom jeziku C koji određuje koje od navedenih osoba su poznate. Program sa standardnog ulaza učitava parove celih brojeva, koji predstavljaju identifikatore osoba, pri čemu broj -1 označava kraj unosa. Par vrednosti **a b** znači da osoba **a** poznaje osobu **b**, ali ne važi obrnuto. Validni identifikatori osoba su u opsegu [0, N], pri čemu se najpre broj N učitava sa standardnog ulaza. Osoba **a** je poznata ukoliko važi da barem polovina osoba od svih osoba poznaje osobu **a**, i da osoba **a** ne poznaje nijednu drugu osobu. Program realizovati prema sledećim stavkama:

1. Napisati funkciju `int** readAcquaintances(int *n);` koja na osnovu podataka sa standardnog ulaza računa i vraća pokazivač na matricu koja sadrži informacije o poznanstvima. Svaki red matrice odgovara jednoj osobi i sastoji se od dva elementa. Prvi element predstavlja broj osoba koje ta osoba poznaje, dok drugi element predstavlja broj osoba koje poznaju tu osobu.
2. Napisati funkciju `void findPopular(int **m, int n);` koja na osnovu matrice poznanstva nalazi i ispisuje identifikatore svih poznatih osoba u jednom redu, po rastućem redosledu identifikatora, odvojene jednim blanko znakom.

Korišćenjem prethodno realizovanih funkcija napisati glavni program koji treba da pročita sve parove celih brojeva, a zatim odredi i ispiše sve poznate osobe u jednom redu, po rastućem redosledu identifikatora, odvojene jednim blanko znakom.

Primer ulaza:	Primer izlaza:
6 0 2 3 2 1 0 3 4 1 4 5 2 3 1 0 4 5 4 -1	4

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

#define CALLOC( ptr, cnt, size ){\
    if(!(ptr = calloc(cnt, size))) {\
        exit(0);\
    }\
}


void freeSpace ( int **acq, int n ) {
    for ( int i = 0; i <= n; i++ ) {
        free ( acq[ i ] );
    }
    free ( acq );
}

int **readAcquaintances ( int *n ) {
    scanf ( "%d", n );
    if ( *n < 0 ) {
        // exit ( 0 );
        return NULL;
    }

    int **acq;
    CALLOC( acq, ( *n + 1 ), sizeof ( *acq ));

    for ( int i = 0; i <= *n; i++ ) CALLOC( acq[ i ], 2, sizeof ( **acq ));

    int a, b;
    while ( 1 ) {
        scanf ( "%d", &a );

        if ( a == -1 ) {
            break;
        }

        scanf ( "%d", &b );
        acq[ a ][ 0 ]++;
        acq[ b ][ 1 ]++;
    }

    return acq;
}

void findPopular ( int **m, int n ) {
    for ( int i = 0; i <= n; i++ ) {
        if ( !m[ i ][ 0 ] && m[ i ][ 1 ] >= ( n + 1 ) / 2. ) {
            printf ( "%d ", i );
        }
    }
}


int main ( ) {
    int n;
    int **acquaintances = readAcquaintances ( &n );
    findPopular ( acquaintances, n );
    for ( int i = 0; i <= n; i++ ) {
        free ( acquaintances[ i ] );
    }
    free ( acquaintances );
    return 0;
}
```

Zadatak 2

Napisati program na programskom jeziku C koji vrši obradu podataka koji se tiču evidencije broja obolelih od korona virusa po državama. Podaci se nalaze u datoteci **evidencija.txt**, za svaku državu u posebnom redu u sledećem formatu: naziv države (niz od najviše 30 znakova koji može sadržati jedan ili više blanko znakova), broj dana za koji se vodila evidencija (ceo broj), a nakon toga niz celih brojeva koji predstavljaju broj novoobolelih po danu, za svaki dan vođenja evidencije. Svi podaci, sem naziva države, su razdvojeni jednim blanko znakom. Naziv države je razdvojen uspravnom crtom od ostatka podataka. Broj redova u datoteci je nepoznat. Program prvo treba da učita informacije iz ulazne datoteke, a zatim da za svaku državu odredi najveći broj uzastopnih dana u kojima je broj novoobolelih bio u porastu. Program nakon toga treba da sortira države po najvećem broju uzastopnih dana u kojima je broj novoobolelih bio u porastu opadajuće, a u slučaju dve ili više država sa jednakim brojem leksikografski rastuće prema nazivu. Program treba zatim da u datoteku **statistika.txt** upiše podatke u sortiranom poretku po prethodno navedenim kriterijumima po formatu: naziv države, broj dana. Svi podaci treba da budu razdvojeni jednim blanko znakom. Pretpostaviti da postoji datoteka **type.h** u kojoj se nalaze sledeće definicije tipova:

```
#ifndef TYPE_H
#define TYPE_H

typedef struct data {
      char name[31];
      int numberOfDays;
      int *days;
} TData;

typedef struct node {
      TData *data;
      struct node *next;
} TNode;

#endif
```

Program realizovati prema sledećim stavkama:

1. Napisati funkciju **TNode* readNode(FILE *input);** koja iz ulazne datoteke **input** čita jedan red na osnovu kojeg formira jedan član liste čiju adresu vraća kao povratnu vrednost funkcije. Ukoliko je datoteka pročitana do kraja funkcija kao povratnu vrednost treba da vrati **NULL**.
2. Korišćenjem prethodno realizovane funkcije, napisati funkciju **TNode* readList(FILE *input);** koja na osnovu sadržaja ulazne datoteke **input** formira listu u kojoj svaki čvor sadrži informacije iz jednog reda datoteke.
3. Napisati funkciju **int consecutiveGrowth(TData *data);** koja na osnovu argumenta **data** računa broj uzastopnih dana u kojima je broj novoobolelih bio u porastu.
4. Korišćenjem prethodno realizovanih funkcije, napisati funkciju **void sort(TNode *head);** koja sortira listu na koju pokazuje pokazivač **head** rastuće prema najvećem broj uzastopnih dana u kojima je broj novoobolelih bio u porastu, a u slučaju dve ili više država sa jednakim brojem leksikografski rastuće.
5. Korišćenjem prethodno realizovanih funkcije, napisati funkciju **void print(TNode *head, FILE *output);** koja ispisuje listu na koju pokazuje pokazivač **head** u datoteku **output** tako što ispisuje naziv države i najvećem broj uzastopnih dana u kojima je broj novoobolelih bio u porastu razdvojene jednim blanko znakom.

Korišćenjem prethodno realizovanih funkcija napisati glavni program koji formira listu na osnovu ulazne datoteke, izvršava zahtevanu obradu i ispisuje rezultat u izlaznu datoteku.

Datoteka evidencija.txt:	Datoteka statistika.txt
Spanija|11 1 20 50 300 250 200 350 400 500 600 550 Sjedinjene Americke Drzave|6 3 100 1000 2500 3000 2700 Italija|10 5 70 200 180 190 250 300 700 1000 900 Velika Britanija|4 5 20 10 70	Italija 6 Sjedinjene Americke Drzave 5 Spanija 5 Velika Britanija 2

```c
#include "type.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define ALLOC( ptr, bytes ) {\
    if(!(ptr = malloc(bytes))) {\
        exit(0);\
    }\
}

#define FOPEN( ptr, name, mode ) {\
    if(!(ptr = fopen(name, mode))) {\
        exit(0);\
    }\
}

void freeData ( TNode *head ) {
    TNode *old;
    while ( head ) {
        old  = head;
        head = head->next;
        free ( old->data->days );
        free ( old->data );
        free ( old );
    }
}

TNode *readNode ( FILE *input ) {
    TData data, *p_data;
    TNode *node;
    if ( fscanf ( input, "%[^|]|%d", data.name, &data.numberOfDays ) == 2 ) {
        ALLOC( data.days, data.numberOfDays * sizeof ( int * ));
        for ( int i = 0; i < data.numberOfDays; i++ ) {
            fscanf ( input, "%d", data.days + i );
        }
        // fgetc(input);//get newline
        fscanf ( input, "%*[^\n]\n" );
        ALLOC( node, sizeof ( *node ));
        ALLOC( p_data, sizeof ( *p_data ));
        *p_data = data;
        node->data = p_data;
        node->next = NULL;
        return node;
    } else { return NULL; }
}
```

```
TNode *readList ( FILE *input ) {
    TNode *node, *first = NULL, *last = first;
    while ( node = readNode ( input )) {
        node->next = NULL;
        if ( last ) {
            last->next = node;
        } else {
            first = node;
        }
        last       = node;
    }
    return first;
}

int consecutiveGrowth ( TData *data ) {
    int       maxDays = 1, currDays = 1;
    for ( int i       = 1; i < data->numberOfDays; i++ ) {
        if ( data->days[ i - 1 ] < data->days[ i ] ) {
            currDays++;
        } else {
            if ( currDays > maxDays ) {
                maxDays = currDays;
            }
            currDays = 1;
        }
    }
    if ( currDays > maxDays ) {
        maxDays = currDays;
    }
    return maxDays;
}

void sort ( TNode *head ) {
    for ( ; head; head = head->next ) {
        for ( TNode *next = head->next; next; next = next->next ) {
            int daysH = consecutiveGrowth ( head->data ),
                daysN = consecutiveGrowth ( next->data );
            if ( daysN > daysH || daysN == daysH && strcmp ( next->data->name, head-
>data->name ) < 0 ) {
                TData data = *head->data;
                *head->data = *next->data;
                *next->data = data;
            }
        }
    }
}

void print ( TNode *head, FILE *output ) {
    for ( ; head; head = head->next ) {
        fprintf ( output, "%s %d\n", head->data->name, consecutiveGrowth ( head->data ));
    }
}
```

```c
int main ( ) {
    FILE *input, *output;
    FOPEN( input, "evidencija.txt", "r" );
    TNode *head = readList ( input );
    fclose ( input );

    FOPEN( output, "statistika.txt", "w" );

    sort ( head );
    print ( head, output );

    fclose ( output );
    freeData ( head );
    return 0;
}
```

Ispit iz Programiranja 2

Šifra zadatka

60 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.

PITANJA

Pitanje

Na nekom računaru realni brojevi se predstavljaju na 9 bita u formatu **seeeemmmm**, gde je **s** bit za predznak broja, **eeee** biti eksponenta u kodu sa viškom 7, a **mmmm** biti normalizovane mantise sa skrivenim bitom ($1 \leq M < 2$). Celi brojevi se predstavljaju na širini od 8 bitova u komplementu dvojke. Izgled celobrojne promenljive, smeštene u lokaciju A, je $3D_{16}$, a izgled realne promenljive, smeštene u lokaciju B, je 225_8. Ukoliko se izvrši operacija C = A - B, kako će izgledati predstava rezultujućeg realnog broja u lokaciji C nakon izračunavanja te operacije? Sva zaokruživanja obavljati prema pravilima ANSI/IEEE standarda za realne brojeve.

Ⓡ) 314_8 (Ⓡ) 011001011_2 Ⓡ) $1CB_{16}$

Pitanje

Šta ispisuje sledeći program napisan na programskom jeziku C?

```
#include <stdio.h>
void main() {
   unsigned int i=0x18, cnt = 0;
   while(i & (i - 1)) {
      switch (i & 03) {
         case 0: i >>= 1; break;
         case 1: i | (i + 1);
         case 2: i ^= 1; continue;
         default: i += 1;
      }
      cnt++;
   }
   printf("%d", cnt);
}
```

(Ⓡ) 3 Ⓡ) 4 Ⓡ) 5

Pitanje

Šta ispisuje sledeći program napisan na programskom jeziku C ako se putem standardnog ulaza prosledi string **arya**?

```
#include <stdio.h>
#include <string.h>
#define MAX 10
void fun(char *str, char *q) {
   char *p = str, *r = q - 1, t;
   if(q - p <= 0) return;
   while(1) {
      while(*p < *q) p++;
      while(r > p && *r >= *q) r--;
      if(p < r) { t = *p; *p = *r; *r = t; }
      else break;
   }
   t = *q; *q = *p; *p = t;
   fun(str, p - 1);
   fun(p + 1, q);
}
int main(void) {
   char s[MAX];
   scanf("%s", s);
   fun(s, s + strlen(s) - 1);
   printf("%s", s);
}
```

(Ⓡ) aary Ⓡ) aray Ⓡ) aayr

Pitanje

Šta ispisuje sledeći program napisan na programskom jeziku C? Program je pokrenut sledećom komandom: **`./p2.exe mart jun kovid-19 2020`**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main(int argc, char *argv[]) {
   int len = argc - 1, left = 0, right = len;
   char **str = malloc(len * sizeof(*str));
   for (int i = 1, j; i < argc; i++) {
      if(strlen(argv[0]) > strlen(argv[i])) {
         str[--right] = argv[i];
         continue;
      }
      for(j = left - 1; j >= 0; j--) {
         if(strcmp(str[j], argv[i]) <= 0)
            break;
         str[j + 1] = str[j];
      }
      str[j + 1] = argv[i]; left++;
   }
   for (int i = 0; i < len; i++)
      printf("%s ", str[i]);
   free(str);
}
```

(®) kovid-19 2020 jun mart ®) jun 2020 mart kovid-19 ®) 2020 kovid-19 jun mart

Pitanje

Šta radi sledeća funkcija napisan na programskom jeziku C nad jednostruko ulančanom listom celih brojeva? Pretpostaviti da lista nije prazna.

```
typedef struct elem {
    int broj;
    struct elem *sled;
} Elem;
void listFunc(Elem *lst) {
  Elem *i;
  int n = 0, j;
  for (i = lst; i; i = i->sled) n++;
  for (j = 0; j < n; j++)
    for (i = lst; i->sled != NULL; i = i->sled)
      if (i->broj < i->sled->broj) {
        int tmp = i->broj;
        i->broj = i->sled->broj;
        i->sled->broj = tmp;
      }
}
```

®) Obrće redosled elemenata u listi.
(®) Uređuje zadatu listu u nerastućem poretku.
®) Premešta na kraj liste sve elemente koji su veći od svog neposrednog sledbenika.

Ispit iz Programiranja 2

II Deo

90 minuta

Napomene:
a) Prvi zadatak nosi **20** poena, dok drugi nosi **25** poena.

ZADACI

Zadatak 1

Napisati program na programskom jeziku C koji određuje pobednike šahovskog turnira. Program sa standardnog ulaza učitava parove celih brojeva, koji predstavljaju identifikatore učesnika turnira, pri čemu broj -1 označava kraj unosa. Svi učesnici na početku imaju 0 poena. Par vrednosti **`a b`** znači da je učesnik **`a`** pobedio učesnika **`b`**. U tom slučaju učesnik **`a`** dobija 2 poena, dok učesnik **`b`** gubi 1 poen. Validni identifikatori učesnika su u opsegu [0, N], pri čemu se broj N unosi sa standardnog ulaza. Pobednik turnira je učesnik koji ima najviše poena. Program realizovati prema sledećim stavkama:

1. Napisati funkciju **`int** readResults(int *n);`** koja na osnovu podataka sa standardnog ulaza računa i vraća pokazivač na matricu koja sadrži informacije o partijama. Svaki red matrice odgovara jednom učesniku i sastoji se od dva elementa. Prvi element predstavlja broj dobijenih partija, dok drugi element predstavlja broj izgubljenih partija.
2. Napisati funkciju **`void findWinner(int **m, int n);`** koja na osnovu matrice partija nalazi i ispisuje identifikator pobednika turnira i broj poena koje je osvojio. Ukoliko ima više učesnika sa istim maksimalnim rezultatom, ispisati ih u jednom redu, po rastućem redosledu identifikatora, odvojene jednim blanko znakom.

Korišćenjem prethodnih funkcija napisati glavni program koji treba da pročita sve parove celih brojeva, a zatim odredi i ispiše pobednika. Ukoliko više učesnika ima isti maksimalan broj poena ispisati ih sve u jednom redu, po rastućem redosledu identifikatora, odvojene jednim blanko znakom.

Primer ulaza:	Primer izlaza:
5 0 2 3 2 1 0 3 4 1 4 5 2 3 1 0 4 5 4 -1	3 6

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

#define CALLOC( ptr, cnt, size ){\
    if(!(ptr = calloc(cnt, size))) {\
        exit(0);\
    }\
}

void freeSpace ( int **matrix, int n ) {
    for ( int i = 0; i <= n; i++ ) {
        free ( matrix[ i ] );
    }
    free ( matrix );
}

int **readResults ( int *n ) {
    scanf ( "%d", n );
    if ( *n < 0 ) {
        return NULL;
    }

    int **res;
    CALLOC( res, ( *n + 1 ), sizeof ( *res ));

    for ( int i = 0; i <= *n; i++ ) CALLOC( res[ i ], 2, sizeof ( **res ));

    int a, b;
    while ( 1 ) {
        scanf ( "%d", &a );
        if ( a == -1 ) {
            break;
        }
        scanf ( "%d", &b );
        res[ a ][ 0 ]++;
        res[ b ][ 1 ]++;
    }

    return res;
}

void findWinner ( int **m, int n ) {
    int maxRes;

    for ( int i = 0; i <= n; i++ ) {
        if ( !i || maxRes < ( 2 * m[ i ][ 0 ] - m[ i ][ 1 ] )) {
            maxRes = 2 * m[ i ][ 0 ] - m[ i ][ 1 ];
        }
    }

    for ( int i = 0; i <= n; i++ ) {
        if (( 2 * m[ i ][ 0 ] - m[ i ][ 1 ] ) == maxRes ) {
            printf ( "%d %d ", i, maxRes );
        }
    }
}

int main ( ) {
    int n;
    int **results = readResults ( &n );
    findWinner ( results, n );
    freeSpace ( results, n );
    return 0;
}
```

Zadatak 2

Napisati program na programskom jeziku C koji vrši obradu podataka koji se tiču evidencije broja obolelih od korona virusa po državama. Podaci se nalaze u datoteci **evidencija.txt**, za svaku državu u posebnom redu u sledećem formatu: naziv države (niz od najviše 30 znakova koji može sadržati jedan ili više blanko znakova), broj dana za koji se vodila evidencija (ceo broj), a nakon toga niz celih brojeva koji predstavljaju broj novoobolelih po danu, za svaki dan vođenja evidencije. Svi podaci, sem naziva države, su razdvojeni jednim blanko znakom. Naziv države je razdvojen uspravnom crtom od ostatka podataka. Broj redova u datoteci je nepoznat. Program prvo treba da učita informacije iz ulazne datoteke, a zatim da za svaku državu odredi broj „ekstremnih" dana u kojima je broj novoobolelih bio veći od broja novoobolelih u prethodnom i u narednom danu. Prvi i poslednji dan ne mogu biti „ekstremni" dani. Program nakon toga treba da sortira države po broju „ekstremnih" dana opadajuće, a u slučaju dve ili više država sa jednakim brojem leksikografski rastuće prema nazivu. Program treba zatim da u datoteku **statistika.txt** upiše podatke u sortiranom poretku po prethodno navedenim kriterijumima po formatu: naziv države, broj „ekstremnih" dana. Svi podaci treba da budu razdvojeni jednim blanko znakom. Pretpostaviti da postoji datoteka **type.h** u kojoj se nalaze sledeće definicije tipova:

```
#ifndef TYPE_H
#define TYPE_H

typedef struct data {
      char name[31];
      int numberOfDays;
      int *days;
} TData;

typedef struct node {
      TData *data;
      struct node *next;
} TNode;

#endif
```

Program realizovati prema sledećim stavkama:

1. Napisati funkciju **TNode * readNode(FILE *input);** koja iz ulazne datoteke **input** čita jedan red na osnovu kojeg formira jedan član liste čiju adresu vraća kao povratnu vrednost funkcije. Ukoliko je datoteka pročitana do kraja funkcija kao povratnu vrednost treba da vrati **NULL**.
2. Korišćenjem prethodno realizovane funkcije, napisati funkciju **TNode* readList(FILE *input);** koja na osnovu sadržaja ulazne datoteke **input** formira listu u kojoj svaki čvor sadrži informacije iz jednog reda datoteke.
3. Napisati funkciju **int extremeDays(TData *data);** koja na osnovu argumenta **data** računa broj „ekstremnih" dana.
4. Korišćenjem prethodno realizovanih funkcije, napisati funkciju **void sort(TNode *head);** koja sortira listu na koju pokazuje pokazivač **head** rastuće prema broju dobrih „ekstremnih", a u slučaju dve ili više država sa jednakim brojem leksikografski rastuće.
5. Korišćenjem prethodno realizovanih funkcije, napisati funkciju **void print(TNode *head, FILE *output);** koja ispisuje listu na koju pokazuje pokazivač **head** u datoteku **output** tako što ispisuje naziv države i broj „ekstremnih" dana razdvojene jednim blanko znakom.

Korišćenjem prethodno realizovanih funkcija napisati glavni program koji formira listu na osnovu ulazne datoteke, izvršava zahtevanu obradu i ispisuje rezultat u izlaznu datoteku.

Datoteka evidencija.txt:	Datoteka statistika.txt
Spanija|11 1 20 50 300 250 200 350 400 500 600 550 Sjedinjene Americke Drzave|6 3 100 1000 2500 3000 2700 Italija|10 5 70 200 180 190 250 300 700 1000 900 Velika Britanija|4 5 20 10 70	Italija 2 Spanija 2 Sjedinjene Americke Drzave 1 Velika Britanija 1

```c
#include "type.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define ALLOC( ptr, bytes ) {\
    if(!(ptr = malloc(bytes))) {\
        exit(0);\
    }\
}

#define FOPEN( ptr, name, mode ) {\
    if(!(ptr = fopen(name, mode))) {\
        exit(0);\
    }\
}

void freeData ( TNode *head ) {
    TNode *old;

    while ( head ) {
        old  = head;
        head = head->next;
        free ( old->data->days );
        free ( old->data );
        free ( old );
    }
}

TNode *readNode ( FILE *input ) {
    TData data, *p_data;
    TNode *node;

    if ( fscanf ( input, "%[^|]|%d", data.name, &data.numberOfDays ) == 2 ) {
        ALLOC( data.days, data.numberOfDays * sizeof ( int * ));
        for ( int i = 0; i < data.numberOfDays; i++ ) {
            fscanf ( input, "%d", data.days + i );
        }
        fgetc ( input );
        ALLOC( node, sizeof ( *node ));
        ALLOC( p_data, sizeof ( *p_data ));
        *p_data = data;
        node->data = p_data;
        return node;
    } else { return NULL; }
}
```

```
TNode *readList ( FILE *input ) {
    TNode *node, *first = NULL, *last = first;

    while ( node = readNode ( input )) {
        node->next = NULL;
        if ( last ) {
            last->next = node;
        } else {
            first = node;
        }
        last       = node;
    }

    return first;
}

int extremeDays ( TData *data ) {
    int cnt = 0;

    for ( int i = 1; i < data->numberOfDays - 1; i++ ) {
        if ( data->days[ i - 1 ] < data->days[ i ] && data->days[ i ] > data->days[ i + 1
] ) {
            cnt++;
        }
    }

    return cnt;
}

void sort ( TNode *head ) {
    for ( ; head; head = head->next ) {
        for ( TNode *next = head->next; next; next = next->next ) {
            int daysH = extremeDays ( head->data ),
                daysN = extremeDays ( next->data );
            if ( daysN > daysH || daysN == daysH && strcmp ( next->data->name, head-
>data->name ) < 0 ) {
                TData data = *head->data;
                *head->data = *next->data;
                *next->data = data;
            }
        }
    }
}

void print ( TNode *head, FILE *output ) {
    for ( ; head; head = head->next ) {
        fprintf ( output, "%s %d\n", head->data->name, extremeDays ( head->data ));
    }
}

int main ( ) {
    FILE *input, *output;
    FOPEN( input, "evidencija.txt", "r" );
    TNode *head = readList ( input );
    fclose ( input );

    FOPEN( output, "statistika.txt", "w" );

    sort ( head );
    print ( head, output );

    fclose ( output );
    freeData ( head );

    return 0;
}
```