Zadanie | Testy

V súboroch `cisla1.txt` a `cisla2.txt` sa nachádzajú usporiadané postupnosti celých čísel oddelených medzerami. Napíšte program, ktorý spojí tieto dve postupnosti do jednej spoločnej postupnosti do súboru `vysledok.txt` tak, aby výsledná postupnosť bola usporiadaná, a aby obsahovala každé z čísel zo súborov `cisla1.txt` a `cisla2.txt`. Postupnosti môžu byť ľubovoľne dlhé.

Ukážka súboru `cisla1.txt`:

```
2 4 6 8 10 12 14 16
```

Ukážka súboru `cisla2.txt`:

```
-10 -5 0 5 10
```

Ukážka súboru `vysledok.txt`:

```
-10 -5 0 2 4 5 6 8 10 10 12 14 16
```

uloha7-1.c | vysledok.txt | cisla2.txt | cisla1.txt

```c
// uloha7-1.c -- Peter Plevko, 14.11.2019 14:07

#include <stdio.h>

int main()
{
  int ok1, c1, ok2, c2, posun;
  FILE *f1 = fopen("cisla1.txt", "rt");
  FILE *f2 = fopen("cisla2.txt", "rt");
  FILE *fout = fopen("vysledok.txt", "wt");
  ok1 = fscanf(     f1, "%d" , &c1     );
  ok2 = fscanf(     f2, "%d", &c2      );
  while (     ok1==1||ok2==1      ) // ak je platne aspon jedno cislo
  {
    posun = 0;
    if (    ok1==1 && ok2==1    )
      posun = (    c1<=c2    ) ? 1 : 2;
    else
      posun = (    ok1==1    ) ? 1 : 2;

    if (posun == 1)
    {
      fprintf(fout, "%d ", c1);
      ok1 = fscanf(      f1 ,"%d",&c1      );
    }
    else
    {
      fprintf(fout, "%d ", c2);
      ok2 = fscanf(      f2,"%d",&c2      );
    }
  }
  fclose(fout);
  return 0;
}
```

Kompilácia

Štandardný vstup

```
1
```

Štandardný výstup

```
1
```

Zadanie | Testy

Palindróm je reťazec, ktorý je rovnaký ked ho čítame spredu ako keď ho čítame zozadu. Napr. abba je palindrom, podobne abeceba je palindróm, ale abeceda nie je palindrom (aj keď možno symetricky vyzerá).

Napíšte funkciu, ktorá zistí, či je reťazec palindróm.

Naprogramujte funkciu v nasledovnom tvare:

```
// Funkcia zisti, ci retazec str je palindrom. Vrati 1, ak je ret
int je_palindrom(char *str)
{
  // ...
}
```

Reťazec str="aa" je palindróm.
Reťazec str="aaa" je palindróm.
Reťazec str="ab" nie je palindróm.
Reťazec str="abc" nie je palindróm.
Reťazec str="aba" je palindróm.
Reťazec str="abba" je palindróm.

uloha7-2.c

```c
// uloha7-2.c -- Peter Plevko, 13.11.2019 23:44

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

// Funkcia zisti, ci retazec str je palindrom. Vrati 1, ak je retazec palindrom, inak 0.
int je_palindrom(char *str)
{
 int l = 0;
  int h = strlen(str) - 1;

  while (h > l)
  {
    if (str[l++] != str[h--])
    {
      return 0;
    }
  }
return 1;




}

// ukazkovy test (spracovanie vstupu)
int main(void)
{
  char buf[1000];

  // nacitanie vstupu
  while (scanf("%s", buf) > 0)
  {
    if (je_palindrom(buf))
      printf("PALINDROM\n");
    else
      printf("NIE\n");
  }
```

Kompilácia

Štandardný vstup

```
1
```

Štandardný výstup

```
1
```

Zadanie | Testy

Daný je neprázdny reťazec `str` párnej dĺžky a znak `c`. Napíšte funkciu `vloz_do_stredu(char *str, char c)`, ktorá do stredu reťazca `str` vloží znak `c`. Napr. pre `str="ABBA"` a znak `c='+'` bude upravený reťazec `"AB+BA"`. Predpokladajte, že pole znakov `str` má dostatok miesta pre ďalší znak.

Hlavná funkcia `main()` načíta hodnoty `str` a `c` zo vstupu a vypíše upravený reťazec `str`.

Ukážka vstupu:

```
ABBA +
Nira t
```

Výstup pre ukážkový vstup:

```
AB+BA
Nitra
```

uloha7-3.c

```c
// uloha7-3.c -- Peter Plevko, 13.11.2019 23:44

#include <stdio.h>

void vloz_do_stredu(char *str, char c)
{
int i;

  for (i=strlen(str);i>=strlen(str)/2;i--)
  {
    str[i+1]=str[i];
  }
str[strlen(str)/2]=c;

}

int main()
{
  char buf[1000], znak[10];
  while(scanf("%s %s", buf, znak) == 2)
  {
    vloz_do_stredu(buf, znak[0]);
    printf("%s\n", buf);
  }
  return 0;
}
```

Štandardný vstup

```
00 +
0123456789 +
01234567890123456789 +
0+00+0 =
01234+5678901234+56789 =
0123456789+01234567890123456789+0123456789 =
```

Štandardný výstup

Kompilácia

Zadanie | Testy

Napíšte funkciu `odstran_male_pismena(char *str)`, ktorá vo vstupnom reťazci `str` odstráni písmená malej anglickej abecedy. Napr. reťazec `"SlovenskaRepublika"` upraví na `"SR"`.

Hlavná funkcia `main()` načíta hodnoty `str` zo vstupu a vypíše upravený reťazec `str`.

Ukážka vstupu:

```
SlovenskaRepublika
```

Výstup pre ukážkový vstup:

```
SR
```

uloha7-4.c

```c
// uloha7-4.c -- Peter Plevko, 13.11.2019 23:44
#include <stdio.h>
#include <string.h>

#include <stdio.h>
#include <ctype.h>
#include <stdio.h>


void odstran_male_pismena(char* str)
{
    char pole[100];
    // sem napis svoje riesenie
    int i, j = 0;
    for (i = 0; i < strlen(str); i++)
    {
        if (isupper(str[i]) == 0)str[i] = '0';
    }

    for (i = 0; i < strlen(str); i++)
    {
        if (isupper(str[i]) != 0)
        {
            pole[j] = str[i];
            j++;

        }
        pole[j] = 0;
    }

    strcpy(str, pole);


}

int main()
{
```

Kompilácia

Štandardný vstup

```
SlovenskaRepublika
```

Štandardný výstup

```

```

Zadanie | Testy

Daný je neprázdny reťazec `str` obsahujúci anglický text. Napíšte funkciu `char najcastejsi_znak(char *str)`, ktorá ako návratovú hodnotu vráti najčastejšie sa vyskytujúci znak v reťazci str. Uvažujte len znaky malej anglickej abecedy.

Napr. pre reťazec `str = "Dobromilka a Svetlusik sa ucia programovat!"` je najčastejšie sa opakujúce písmeno malej anglickej abecedy písmeno `'a'`, ktoré sa opakuje 6 krát.

Hlavná funkcia `main()` načíta hodnoty `str` zo vstupu a na výstup vypíše výsledok funkcie.

Ukážka vstupu:

```
Dobromilka a Svetlusik sa ucia programovat!
```

Výstup pre ukážkový vstup:

```
a
```

uloha7-5.c

```c
// uloha7-5.c -- Peter Plevko, 13.11.2019 23:44

#include <stdio.h>
#include<string.h>
char najcastejsi_znak(char *str)
{
    int i, len;
    int max = -1;

    int freq[26] = {0};




    len = strlen(str);

    for(i = 0; i < len; i++)
    {
      if(str[i] >= 'a' && str[i] <= 'z')
        freq[str[i] - 'a']++;
      }
    max = 0;
    for(i = 0; i < 26; i++)
    {
        if(freq[max] < freq[i])
        {
            max = i;
        }
    }
 return (max + 'a');
}

int main(void)
{
  char buf[10000];
  while(fgets(buf, 10000, stdin))
    printf("%c\n", najcastejsi_znak(buf));
```

Kompilácia

Štandardný vstup

```
The Slavs arrived in the territory of present-
In the 7th century, they played a significant
In the 10th century, after the dissolution of
In 1241 and 1242, much of the territory was de
```

Štandardný výstup

```

```

Zadanie | Testy

Postupka je reťazec, v ktorom sú písmenká abecedy za sebou, pričom za písmenom z nasleduje opäť a. Postupka je zložená alebo z malých písmen anglickej abecedy, alebo z veľkých písmen anglickej abecedy.

Napíšte funkciu, ktorá pre daný počiatočný znak `z` postupky a jej dĺžku `len` vygeneruje do reťazca postupku dĺžky `len` začínajúci znakom `z`.

Naprogramujte funkciu v nasledovnom tvare:

```
// Funkcia vygeneruje postupku do retazca str: zacinajucu znakom
void postupka(char *str, char z, int len)
{
  // ...
}
```

Pre z='a' a len=3 je postupka: abc
Pre z='c' a len=10 je postupka: cdefghijkl
Pre z='E' a len=5 je postupka: EFGHI
Pre z='x' a len=30 je postupka: xyzabcdefghijklmnopqrstuvwxyza

uloha7-6.c

```c
// uloha7-6.c -- Peter Plevko, 14.11.2019 15:39
#include <stdio.h>
#include <stdlib.h>

// Funkcia vygeneruje postupku do retazca str: zacinajucu znakom z, dlzky len.
void postupka(char *str, char z, int len)
{

  int i;
  for (i = 0; i < len; i++)
  {
    str[i] = z++;
    if (z == 'z' + 1)
      z = 'a';
    if (z == 'Z' + 1)
      z = 'A';
  }
  str[i] = 0;
  return str;


}

// ukazkovy test (spracovanie vstupu)
int main(void)
{
  char z[2], *buf;
  int n;

  // nacitanie vstupu
  while (scanf("%s %d", z, &n) == 2)
  {
    buf = malloc(n + 1); // vyhradim n+1 znakov pre retazec
    postupka(buf, z[0], n);
    printf("%s\n", buf);
    free(buf); // uvolnim vyhradene miesto
  }
```

Kompilácia

Štandardný vstup

Štandardný výstup

Zadanie | Testy

Napíšte funkciu `int strinsert(char *dst, int len, const char *src, int offset)`, ktorá do reťazca `dst` od pozície `offset` vloží kópiu reťazca `src`. Argument `len` určuje počet znakov vyhradených pre pole `dst` (vrátane ukončovacieho znaku `\0`).

Ak nie je možné do vyhradeného miesta reťazec vložiť, funkcia nič nevykoná a vráti 1. Inak (ak je možné do vyhradeného miesta reťazec vložiť), vráti 0. Napr. pre dst:totojeretazec, offset:6, src:druhy volanie strinsert(dst, 50, src, 6) vrati 0 a v reťazci dst bude totojedruhyretazec.

uloha7-7.c

```c
// uloha7-7.c -- Peter Plevko, 14.11.2019 15:39

#include <stdio.h>
#include<string.h>
int strinsert(char *dst, int len, const char *src, int offset)
{
  int dl = strlen(dst), sl=strlen(src);
  if (offset > dl || dl + sl >= len)
    return 1;
  int i;
  for (i = dl+sl; i >= offset+sl; i--)
    dst[i] = dst[i-sl];
  for (i = 0; i < sl; i++)
    dst[offset+i] = src[i];
  return 0;

}

int main()
{
  char dst[100], src[100];
  int len, offset;
  scanf("%s %d %s %d", dst, &len, src, &offset);

  if (strinsert(dst, len, src, offset))
    printf("Nepodarilo sa vlozit retazec.\n");
   else
    printf("%s", dst);
  return 0;
}
```

Kompilácia

Štandardný vstup

```
1
```

Štandardný výstup

```
1
```

Zadanie | Testy

Napíšte funkciu `int strdelete(char *str, int n, int offset)`, ktorá z reťazca `str` od pozície `offset` vymaže `n` znakov.

Ak nie je možné z reťazca od pozície `offset` vymazať `n` znakov, funkcia nič nevykoná a vráti 1. Inak požadované znaky vymaže a vráti 0. Napr. pre str:totojedruhyretazec, strdelete(str, 5, 6) vrati 0 a v reťazci str bude totojeretazec.

uloha7-8.c

```c
// uloha7-8.c -- Peter Plevko, 13.11.2019 23:44

#include <stdio.h>
#include <string.h>
int strdelete(char *str, int n, int offset)
{
char pole[169];
int j=0,i;

if (strlen(str)<n+offset){return 1;}

for (i=offset;i<offset+n;i++)
{
str[i]='0';
}
for (i=0;i<strlen(str);i++)
{
if (str[i]!='0'){ pole[j]=str[i]; j++; }
}
pole[j]=0;
strcpy(str,pole);
return 0;
}


int main()
{
  char str[100];
  int n, offset;
  scanf("%s %d %d", str, &n, &offset);
  if (strdelete(str, n, offset))
    printf("Nepodarilo sa vymazat znaky.\n");
  else
    printf("%s", str);
  return 0;
}
```

Kompilácia

Štandardný vstup

```
totojedruhyretazec 5 6
```

Štandardný výstup

Zadanie | Testy

Napíšte program, ktorý využitím funkcií `strinsert` a `strdelete` z predchádzajúcich úloh (funkcie v tejto úlohe už nemusíte znovu implementovať - sú vložené automaticky), spracuje príkazy na vstupe.

Každý riadku vstupu obsahuje jeden príkaz, nasledovaný parametrami (ak nejaké má). Príkazy môžu byť:

- Príkaz `read` nasledovaným slovom `str` a číslom `max` znamená načítanie aktuálneho slova, s ktorým sa bude pracovať, pričom pole znakov, v ktorom je reťazec reprezentovaný by malo mať vyhradených `max` znakov (vrátane ukončovacieho `\0`). Ak už predtým bol načítaný reťazec `str`, tento sa prepíše novým načítaným slovom. Ak ešte načítaný nebol, `str` obsahuje prázdny reťazec a vyhradených má 50 znakov.
- Príkaz `ins` nasledovaným celým číslom `i` a reťazcom `str2` predstavuje vloženie reťazca `str2` do aktuálneho reťazca `str` od pozície `i` (použitie funkcie `strinsert`).
- Príkaz `del` nasledovaným dvoma celými číslami `i` a `n` znamená vymazanie časti reťazca `str` dlhého `n` znakov od pozície `i` (použitie funkcie `strdelete`).

Po každom načítanom riadku vypíše program do zvlášť riadku aktuálne slovo `str`, s ktorým pracuje. Ak nie je možné niektorý z príkazov vykonať, slovo `str`, ktoré sa nezmenilo, je nasledované jednou medzerou a jednou z chybových správ. Možné chybové správy sú:

- `do retazca nie je mozne vlozit podretazec od zvolenej pozicie` (túto správy vypisujte aj v prípade, že ešte nebol načítaný aktuálny reťazec `str` a príkaz určuje do neho pridávať od inej ako 0-tej pozície),
- `z retazca nie je mozne vymazat znaky` (túto správy vypisujte aj v prípade, že ešte nebol načítaný aktuálny reťazec `str` a príkaz určuje z neho vymazávať).

Ukážka vstupu:

```
ins 7 ahoj
read Programovanie 100
ins 0 VsetciMame
del 10 100
ins 10 Radi
del 0 6
```

Výstup pre ukážkový vstup:

```
 do retazca nie je mozne vlozit podretazec od zvolenej pozicie
Programovanie
VsetciMameProgramovanie
VsetciMameProgramovanie z retazca nie je mozne vymazat znaky
VsetciMameRadiProgramovanie
MameRadiProgramovanie
```

uloha7-9.c

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

// tieto funkcie mozes pouzit vo svojom programe
int strinsert(char *dst, int len, const char *src, int offset);
int strdelete(char *str, int n, int offset);

int main()
{
  char instruction[5];
  char* str = malloc(50 * sizeof(char));
  int strLength = 0;

  while(scanf("%s", instruction) > 0) {

    if(!strcmp(instruction, "read")) {
      scanf("%s", str);
      scanf("%d\n", &strLength);

      str = realloc(str, strLength * sizeof(char));

    } else if(!strcmp(instruction, "ins")) {

      int offset;
      char str2[50];

      scanf("%d %s\n", &offset, str2);

      if((!strLength && offset) || strinsert(str, strLength, str2, offset)) {
        printf("%s do retazca nie je mozne vlozit podretazec od zvolenej pozicie\n", str);
        continue;
      }

    } else if(!strcmp(instruction, "del")) {
      int offset;
      int n;

      scanf("%d %d\n", &offset, &n);

      if(!strLength || strdelete(str, n, offset)) {
        printf("%s z retazca nie je mozne vymazat znaky\n", str);
```

Kompilácia

Štandardný vstup

```
ins 7 ahoj
read Programovanie 40
ins 0 VsetciMame
del 10 100
ins 10 Radi
del 0 6
ins 7 ahoj
ins 0 VsetciMame
del 10 100
ins 10 Radi
del 0 6
read Algoritmy 50
ins 0 VsetciMame
del 10 100
ins 10 Radi
del 0 6
ins 7 ahoj
ins 0 VsetciMame
del 10 100
ins 10 Radi
del 0 6
ins 0 VsetciMame
del 10 100
ins 10 Radi
del 0 6
```

Štandardný výstup