Úlohy na 12. cvičenie

Úloha na precvičenie rekurzie - riešte na papier:

Určte, čo vypíše program, ktorý volá funkciu `r` s hodnotami parametra 3, 4, 5 a 6:

```
void r(int n) {
   if (n > 0) {
      if (n % 2) n--;
      r(n-2);
      printf("%d ", n);
   }
}
```

Výstup pre `r(3)`: _____
Výstup pre `r(4)`: _____
Výstup pre `r(5)`: _____
Výstup pre `r(6)`: _____

Ďalšie úlohy riešte v Turingu:

1. Napíšte program, ktorý načíta celé číslo `n` nasledované znakom konca riadku. Potom načíta postupnosť `n` celých čísel, každé nalsedované znakom konca riadku. Program určí, či načítaná postupnosť čísel je správna. Postupnosť je správna, ak:
   a) Prvé číslo je z rosahu `<0, 10>`
   b) Pre každé `i`-te číslo (`i∈<2,n>`) platí, že nie je väčšie ako dvojnásobok predchádzajúceho (`i-1`)-ho čísla, ani menšie ako polovica predchádzajúceho (`i-1`)-ho čísla.

   Ak je postupnosť správna, vypíše program správu `Postupnost je spravna` a odriadkuje, inak vypíše `Postupnost nie je spravna` a odriadkuje.
   Ukážkový vstup:
   `3↵`
   `5↵`
   `7↵`
   `9↵`
   Ukážkový výstup:
   `Postupnost je spravna↵`

2. Napíšte program, na zisťovanie reverzného čísla. Program načíta zo vstupu číslo `x` ukončené znakom konca riadku. V programe použite funkciu `long reverzne_cislo(long x)` ktorá vráti reverzné číslo k číslu x. Výstupom programu bude vrátené reverzné číslo.
   Program rozšírte tak, že bude načítať všetky čísla zo vstupu. Počet čísel na vstupe nie je známy pred spustením programu. Využite návratovú hodnotu funkcie `scanf`. Ku každému načítanému číslu vytvorí reverzné číslo a navyše zistí, či načítané číslo je

palindróm a vypíše správu: `Cislo X je palindrom`, alebo `Cislo X nie je palindrom`, kde X je zisťované číslo. Správa je nasledovaná znakom konca riadku.
Ukážkový vstup:
```
12345
12321
```
Ukážkový vstup:
```
54321
Cislo 12345 nie je palindrom
12321
Cislo 12321 je palindrom
```

3. Napíšte funkciu `int parne(int x[], int pocetx, int y[])`, ktorá skopíruje všetky párne čísla z poľa `x` do poľa `y` v poradí v akom sa nachádzajú v poli `x` a vráti počet prvkov poľa `y`. Argument `pocetx` určuje počet prvkov poľa `x`.
Môžete predpokladať, že argument `y`, bude mať dostatočnú veľkosť pre všetky párne prvky.
Ukážka volania:
```
x = {4, 7, 1, 3, 2, 5, 6}
pocetx = 7
pocety = parne(x, pocetx, y);  // volanie funkcie
pocety: 3                      // vypis vysledku
y: {4, 2, 6}
```

4. 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
```

5. 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.

6. 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.

7. 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
```

6. Doplňte funkciu `zasifruj(char sprava[], char kluc[], char sifra[])`, tak, aby zašifrovala správu `sprava` kľúčom `kluc` a výsledok uložila do poľa `sifra`. Predpokladajte, že reťazce v poliach `sprava` a `kluc` pozostávajú len z malých písmen.

Kľúč je reťazec, ktorý sa pripočítava k textu správy, pričom ak je kľúč krátky, znova sa opakuje. Pripočítanie písmena kľúča predstavuje posun písmena správy o hodnotu danú písmenom kľúča tak, že hodnota `'a'` predstavuje posun o 0 pozícií, hodnota `'b'` o 1 pozíciu, … až `'a'` o 25 pozícií. Napríklad zašifrovanie správy `aaaaaabbbzzzxx` pomocou kľúča `abc` prebieha nasledovne:

```
aaaaaabbbzzzxx
abcabcabcabcab
--------------
abcabcbcdzabxy
```

Výsledkom je teda šifra `abcabcbcdzab`. Vo funkcii sa na prístup k prvkom poľa používajú ukazovatele.

```
void zasifruj(char sprava[], char kluc[], char sifra[]) {
   char *t, *k, *s;
   int dlzka;
   _________     // t nastavi na zaciatok sprava
   _________     // k nastavi na zaciatok kluc
   _________     // s nastavi na zaciatok sifra
   dlzka = strlen(kluc);
   while(_____________________ ) {  // pokym sa t nachadza v
                                    // ramci nacitaneho retazca
      *s = ___________________________     // zasifruj
      t++;
      s++;
      k++;
      if(k>=kluc+dlzka) _________; // ak k presiahne retazec v
                                   // premennej kluc
   }
   *s = '\0';    // ukoncenie retazca
}
```

7. Doplňte chýbajúce časti v rekurzívnej funkcii `int palindrom(char s[], int z, int k)`, ktorá dostane ako argument reťazec `s`, index prvého znaku `z` a index posledného znaku `k` a vráti hodnotu 1, ak je reťazec palindrómom, inak vráti hodnotu 0.

```
int palindrom(char s[], int z, int k) {
   if(_____)     // jednoprvkovy alebo prazdny retazec
      return 1;
   else if (_________) // prvy a posledny znak sa nerovnaju
      return 0;
   else
      return ____________________;  // rekurzivne volanie
}
```