Corso di Algoritmi e Strutture Dati (IN110) – Prof. Marco Liverani – a.a. 2023/2024

Esame scritto del 8 Febbraio 2024 (Appello B)

Si richiede di risolvere entrambi gli esercizi riportando una codifica in linguaggio C completa dei due programmi. Nel caso in cui non si riesca a completare entrambi gli esercizi si suggerisce di riportare almeno la codifica in C delle funzioni principali o una loro pseudo-codifica. È possibile consultare libri e appunti personali, ma non scambiare libri o appunti con altri studenti. I compiti che presenteranno evidenti ed anomale "similitudini" saranno annullati. La prova scritta ha una durata di tre ore, durante le quali non è consentito allontanarsi dall'aula, se non dopo aver consegnato il compito.
Deve essere consegnata solo la "bella copia" del compito scritto; su ciascun foglio deve essere riportato il ***nome****, il* ***cognome*** *e il* ***numero di matricola*** *(o un altro codice identificativo di fantasia) dello studente.*

Esercizio n. 1

Letto in input un numero $n > 0$ costruire un array A di n numeri interi casuali in $\{0, \ldots, 9\}$. Leggere in input un array B di $n-1$ caratteri (una stringa) composto solo dai simboli '+' e '–'. Stampare gli elementi dell'array A alternandoli con quelli dell'array B e calcolare il risultato dell'espressione aritmetica ottenuta alternando i numeri di A con gli operatori di B.

Esempio Siano $n = 6$; si considerino i seguenti array:

$$\begin{aligned} A &= (5, 2, 4, 1, 8, 2) \\ B &= (\text{'+'}, \text{'+'}, \text{'-'}, \text{'-'}, \text{'+'}) \end{aligned}$$

L'espressione aritmetica da calcolare è pertanto $5 + 2 + 4 - 1 - 8 + 2$ il cui risultato è 4.

Soluzione

```c
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#define MAX 50

int generaNumeri(int A[]) {
  int i, n;
  printf("Numero di elementi: ");
  scanf("%d", &n);
  for (i=0; i<n; i++)
     A[i] = rand() % 10;
  return n;
}

void leggiOperatori(char B[], int n) {
  printf("Inserisci %d operatori di somma e sottrazione: ", n-1);
  scanf("%s", B);
  return;
}

void stampaEspressione(int A[], char B[], int n, int s) {
  int i;
  for (i = 0; i < n-1; i++)
    printf("%d %c ", A[i], B[i]);
```

```
25    printf("%d = %d\n", A[i], s);
26    return;
27  }
28
29  int calcolaEspressione(int A[], char B[], int n) {
30    int s = A[0], i;
31    for (i = 1; i < n; i++) {
32      if (B[i-1] == '+')
33        s = s + A[i];
34      else
35        s = s - A[i];
36    }
37    return s;
38  }
39
40  int main(void) {
41    int A[MAX], n, s;
42    char B[MAX];
43    srand((unsigned)time(NULL));
44    n = generaNumeri(A);
45    leggiOperatori(B, n);
46    s = calcolaEspressione(A, B, n);
47    stampaEspressione(A, B, n, s);
48    return 0;
49  }
```

Esercizio n. 2

Letti in input due numeri interi $n, m > 0$, con $n > m$, acquisire in input due sequenze di n e m caratteri e memorizzarli in due liste L_1 e L_2 rispettando l'ordine con cui sono stati acquisiti. Verificare che la lista L_2 sia un suffisso di L_1.

Esempio Siano $n = 9$, $m = 4$. Consideriamo le seguenti liste di caratteri:

$$\begin{aligned} L_1 &: \mathtt{P} \to \mathtt{a} \to \mathtt{p} \to \mathtt{e} \to \mathtt{r} \to \mathtt{o} \to \mathtt{t} \to \mathtt{t} \to \mathtt{o} \\ L_2 &: \mathtt{o} \to \mathtt{t} \to \mathtt{t} \to \mathtt{o} \\ L_2' &: \mathtt{p} \to \mathtt{e} \to \mathtt{r} \to \mathtt{o} \end{aligned}$$

La lista L_2 è un suffisso di L_1, mentre la lista L_2' non è un suffisso di L_1.

Soluzione

```c
#include <stdlib.h>
#include <stdio.h>
#define MAX 50

struct nodo {
  char c;
  struct nodo *next;
};

void stampaLista(struct nodo *p) {
  while (p != NULL) {
    printf("%c --> ", p->c);
    p = p->next;
  }
  printf("null\n");
  return;
}

struct nodo *leggiLista(int *x) {
  struct nodo *p, *primo = NULL, *ultimo = NULL;
  int i, k;
  printf("Numero di elementi: ");
  scanf("%d", &k);
  *x = k;
  printf("Inserisci %d caratteri della lista: ", k);
  fflush(stdin);
  for (i=0; i<k; i++) {
    p = malloc(sizeof(struct nodo));
    scanf("%c", &p->c);
    if (primo == NULL) {
      primo = p;
    } else {
      ultimo->next = p;
    }
    ultimo = p;
    p->next = NULL;
  }
  return primo;
}
```

```
40
41 int suffisso(struct nodo *L1, int n, struct nodo *L2, int m) {
42   int i, flag = 1;
43   struct nodo *p, *q;
44   p = L1;
45   for (i=0; i<n-m; i++)
46     p = p->next;
47   q = L2;
48   while (p != NULL && q != NULL && flag == 1) {
49     if (p->c == q->c) {
50       p = p->next;
51       q = q->next;
52     } else {
53       flag = 0;
54     }
55   }
56   return flag;
57 }
58
59 int main(void) {
60   struct nodo *L1, *L2;
61   int n, m;
62   L1 = leggiLista(&n);
63   L2 = leggiLista(&m);
64   stampaLista(L1);
65   stampaLista(L2);
66   if (suffisso(L1, n, L2, m))
67     printf("L2 e' un suffisso di L1\n");
68   else
69     printf("L2 NON e' un suffisso di L1\n");
70   return 0;
71 }
```