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

Esame scritto del 18 Gennaio 2023 (Appello A)

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

Letti in input tre interi positivi n, x e y (con $x < y$), costruire un array A con n numeri interi casuali compresi tra x e y (estremi inclusi). Costruire un secondo array B contenente tutti i numeri dell'insieme $\{x, x+1, x+2, \ldots, y\}$ non contenuti in A. Stampare in output A e B.

Esempio Siano $n = 15$, $x = 5$ e $y = 20$ e sia A il seguente array di numeri casuali costruito come richiesto dall'esercizio: $A = (18, 11, 10, 16, 11, 7, 16, 10, 5, 11, 9, 13, 14, 8, 7)$. Allora il vettore B è il seguente: $B = (6, 12, 15, 17, 19, 20)$.

Soluzione

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

void generaArray(int A[], int n, int x, int y) {
  int i;
  srand((unsigned)time(NULL));
  for (i=0; i<n; i++)
     A[i] = rand() % (y - x) + x;
  return;
}

void stampaArray(int A[], int n) {
  int i;
  for (i=0; i<n; i++)
    printf("%d ", A[i]);
  printf("\n");
  return;
}

int contiene(int A[], int n, int z) {
  int i, r=0;
  for (i=0; i<n && r == 0; i++)
    if (A[i] == z)
      r = 1;
  return r;
}
```

```
29
30 int costruisciArray(int A[], int B[], int n, int x, int y) {
31   int i, j=0;
32   for (i=x; i<=y; i++) {
33     if (!contiene(A, n, i)) {
34       B[j] = i;
35       j = j+1;
36     }
37   }
38   return j;
39 }
40
41 int main(void) {
42   int A[MAX], B[MAX], n, m, x, y;
43   printf("Insersci n, x e y: ");
44   scanf("%d %d %d", &n, &x, &y);
45   generaArray(A, n, x, y);
46   m = costruisciArray(A, B, n, x, y);
47   stampaArray(A, n);
48   stampaArray(B, m);
49   return(0);
50 }
```

Esercizio n. 2

Letti in input tre interi positivi n, m, k, generare due liste L_1 e L_2, rispettivamente di n e m elementi, composte da numeri interi casuali in $\{0, 1, 2, \ldots, k\}$. Stampare le due liste. Contare quante volte ogni elemento di L_2 è presente in L_1 e memorizzare il numero di occorrenze in un campo dei nodi della lista L_2. Eliminare i nodi di L_2 che non sono presenti in L_1. Stampare la lista L_2 e il numero di occorrenze di ciascun elemento.

Esempio Siano $n = 10$, $m = 6$ e $k = 5$. Consideriamo le seguenti liste di numeri casuali:

$$\begin{aligned} L_1 &: 3 \to 2 \to 5 \to 0 \to 2 \to 0 \to 5 \to 2 \to 3 \to 5 \\ L_2 &: 1 \to 5 \to 0 \to 5 \to 1 \to 4 \end{aligned}$$

Allora la lista L_2 viene modificata come segue:

$$L_2 : (5,3) \to (0,2) \to (5,3)$$

Soluzione

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

struct nodo1 {
  int info;
  struct nodo1 *next;
};

struct nodo2 {
  int info, ripe;
  struct nodo2 *next;
};

int main(void) {
  int i, n, m, k, c;
  struct nodo1 *p, *primo1 = NULL;
  struct nodo2 *q, *primo2 = NULL, *q1 = NULL;
  scanf("%d %d %d", &n, &m, &k);
  for (i=0; i<n; i++) {
    p = malloc(sizeof(struct nodo1));
    p->info = rand() % k;
    p->next = primo1;
    printf("%d --> ", p->info);
    primo1 = p;
  }
  printf("NULL\n");
  for (i=0; i<m; i++) {
    q = malloc(sizeof(struct nodo2));
    q->info = rand() % k;
    q->ripe = 0;
    q->next = primo2;
    printf("%d --> ", q->info);
    primo2 = q;
  }
  printf("NULL\n");
  q = primo2;
```

```
  while (q != NULL) {
    c = 0;
    p = primo1;
    while (p != NULL) {
      if (p->info == q->info)
        c = c+1;
      p = p->next;
    }
    if (c > 0) {
      q->ripe = c;
      q1 = q;
      q = q->next;
    } else {
      if (q1 == NULL) {
        primo2 = q->next;
        free(q);
        q = primo2;
      } else {
        q1->next = q->next;
        free(q);
        q = q1->next;
      }
    }
  }
  q = primo2;
  while (q != NULL) {
    printf("%d %d volte ", q->info, q->ripe);
    q = q->next;
  }
  printf("\n");
  return 0;
}
```