Corso di Algoritmi e Strutture Dati (IN110) – Prof. Marco Liverani – a.a. 2024/2025

Seconda prova di esonero – 13 gennaio 2025

Risolvere i seguenti problemi proponendo, per ciascun esercizio, la codifica in linguaggio C di un programma completo. La prova dura tre ore, durante le quali non è possibile allontanarsi dall'aula, se non dopo aver consegnato l'elaborato scritto. Per superare la prova di esonero è necessario ottenere almeno 15 punti; tuttavia affinché le prove di esonero siano valide è necessario che la media dei voti del primo e del secondo esonero sia maggiore o uguale a 18/30. È possibile utilizzare libri e appunti personali, senza scambiarli con altri studenti. I compiti che presenteranno evidenti ed anomale "similitudini" saranno annullati.

Esercizio n. 1

Letti in input tre numeri interi $n, a, b > 0$, con $a < b$, costruire una lista L_1 composta da n numeri interi casuali compresi tra a e b (estremi inclusi). Stampare in output la lista L_1. Costruire una seconda lista L_2 composta, per ogni elementi x di L_1, da x elementi con valore 1, se x è dispari e x elementi di valore 0 se x è pari. Stampare la lista L_2.

Esempio Siano $n = 5$, $a = 1$ e $b = 4$ i tre numeri interi letti in input. Supponiamo che sia stata generata la seguente lista di numeri casuali in $\{a, a+1, \ldots, b\}$:

$$L_1 = 3 \to 2 \to 1 \to 3 \to 2$$

Allora la lista L_2 sarà composta come segue:

$$L_2 = 1 \to 1 \to 1 \to 0 \to 0 \to 1 \to 1 \to 1 \to 1 \to 0 \to 0$$

Soluzione

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

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

void stampaLista(struct nodo *p) {
  while (p != NULL) {
    printf("%d --> ", p->info);
    p = p->next;
  }
```

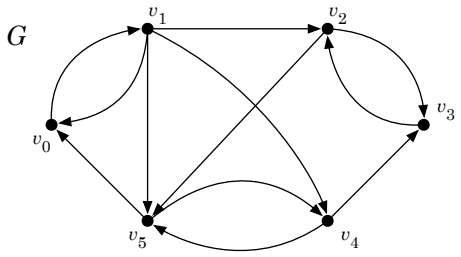
```
15    printf("NULL\n");
16    return;
17  }
18
19  struct nodo *listaCasuale(void) {
20    struct nodo *p, *primo = NULL;
21    int i, n, a, b;
22    printf("Numero di elementi: ");
23    scanf("%d", &n);
24    printf("Inserisci a e b: ");
25    scanf("%d %d", &a, &b);
26    for (i=0; i<n; i++) {
27      p = malloc(sizeof(struct nodo));
28      p->next = primo;
29      p->info = rand() % (b - a + 1) + a;
30      primo = p;
31    }
32    return(primo);
33  }
34
35  struct nodo *costruisciLista(struct nodo *L1) {
36    struct nodo *L2 = NULL, *p, *q, *u = NULL;
37    int i;
38    p = L1;
39    while (p != NULL) {
40      for (i = 0; i < p->info; i++) {
41        q = malloc(sizeof(struct nodo));
42        q->info = p->info % 2;
43        q->next = NULL;
44        if (u == NULL)
45          L2 = q;
46        else
47          u->next = q;
48        u = q;
49      }
50      p = p->next;
51    }
52    return(L2);
53  }
54
55  int main(void) {
56    struct nodo *L1, *L2;
57    srand((unsigned)time(NULL));
58    L1 = listaCasuale();
59    L2 = costruisciLista(L1);
60    stampaLista(L1);
61    stampaLista(L2);
```

```
  return(0);
}
```

Esercizio n. 2

Leggere in input le liste di adiacenza di un grafo orientato $G = (V, E)$ con n vertici. Stampare in output le liste di adiacenza del grafo. Letti in input due vertici $u, v \in V(G)$, verificare se hanno lo stesso insieme di vertici adiacenti ($z \in N(u) \iff z \in N(v)$).

Esempio Si consideri il grafo rappresentato in figura. I vertici v_2 e v_4 hanno lo stesso insieme di vertici adiacenti $N(v_2) = N(v_4) = \{v_3, v_5\}$.



Soluzione

```
#include <stdlib.h>
#include <stdio.h>
#define MAX 100

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

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

void stampaGrafo(struct nodo *G[], int n) {
  int i;
  for (i=0; i<n; i++) {
```

```
22      printf("%2d: ", i);
23      stampaLista(G[i]);
24    }
25    return;
26  }
27
28  struct nodo *leggiLista(void) {
29    struct nodo *p, *primo = NULL;
30    int i, n;
31    printf("Numero di elementi: ");
32    scanf("%d", &n);
33    printf("Elementi della lista: ");
34    for (i=0; i<n; i++) {
35      p = malloc(sizeof(struct nodo));
36      scanf("%d", &p->info);
37      p->next = primo;
38      primo = p;
39    }
40    return primo;
41  }
42
43  int leggiGrafo(struct nodo *G[]) {
44    int i, n;
45    printf("Numero di vertici del grafo: ");
46    scanf("%d", &n);
47    for (i=0; i<n; i++) {
48      printf("Lista di adiacenza del vertice %d\n", i);
49      G[i] = leggiLista();
50    }
51    return n;
52  }
53
54  int adiacenti(struct nodo *G[], int u, int v) {
55    struct nodo *p;
56    int rc;
57    p = G[u];
58    while (p != NULL && p->info != v)
59      p = p->next;
60    if (p == NULL)
61      rc = 0;
62    else
63      rc = 1;
64    return rc;
65  }
66
67  int stessiVicini(struct nodo *G[], int u, int v) {
68    struct nodo *p, *q;
```

```
69    int rc;
70    p = G[u];
71    while (p != NULL && adiacenti(G, v, p->info))
72      p = p->next;
73    q = G[v];
74    while (p == NULL && q != NULL && adiacenti(G, u, q->info))
75      q = q->next;
76    if (p == NULL && q == NULL)
77      rc = 1;
78    else
79      rc = 0;
80    return rc;
81  }
82
83  int main(void) {
84    struct nodo *G[MAX];
85    int n, u, v;
86    n = leggiGrafo(G);
87    stampaGrafo(G, n);
88    printf("Inserisci due vertici: ");
89    scanf("%d %d", &u, &v);
90    if (stessiVicini(G, u, v))
91      printf("i vertici %d e %d hanno gli stessi vicini\n", u, v);
92    else
93      printf("i vertici %d e %d NON hanno gli stessi vicini\n", u, v);
94    return 0;
95  }
```