Corso di Informatica 1 (IN110) – Prof. Marco Liverani – a.a. 2012/2013

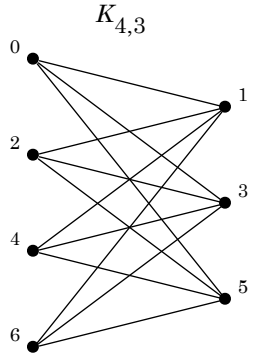
Seconda prova di esonero – 8 gennaio 2013

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

Letto in input un numero intero $n > 1$, costruire le liste di adiacenza del grafo bipartito completo ottenuto considerando come insieme dei vertici $V = \{0, 1, 2, \ldots, n-1\}$ e la partizione di V data dal sottoinsieme di elementi pari e di elementi dispari. Stampare le liste di adiacenza del grafo.

Esempio Sia $n = 7$; il grafo $K_{4,3}$ di cui devono essere costruite le liste di adiacenza è rappresentato nella figura seguente:



Soluzione

```c
#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++) {
      printf("G[%d]: ", i);
      stampaLista(G[i]);
   }
   return;
}

void aggiungiSpigolo(struct nodo *G[], int u, int v) {
   struct nodo *p;
   p = malloc(sizeof(struct nodo));
   p->info = v;
   p->next = G[u];
   G[u] = p;
   return;
}

void costruisciBipartito(struct nodo *G[], int n) {
   int i, j;
   for (i=0; i<n; i++)
      G[i] = NULL;
   for (i=0; i<n; i++) {
      for (j=i+1; j<n; j=j+2) {
         aggiungiSpigolo(G, i, j);
         aggiungiSpigolo(G, j, i);
      }
   }
   return;
}

int main(void) {
   struct nodo *G[MAX];
   int n;
   printf("Numero di vertici: ");
   scanf("%d", &n);
   costruisciBipartito(G, n);
   stampaGrafo(G, n);
   return(0);
}
```

Esercizio n. 2

Letto in input il numero intero $n > 0$, costruire una lista di n numeri naturali casuali minori di 100. Stampare la lista. Costruire una seconda lista con tutti i numeri naturali da 0 a 99 in ordine crescente, non contenuti nella prima lista. Stampare la seconda lista.

Esempio Sia $n = 5$ e supponiamo che la prima lista, costituita da n numeri casuali minori di 10 sia la seguente:

$$27 \to 64 \to 3 \to 50 \to 82$$

Allora la seconda lista costruita dal programma sarà la seguente:

$$0 \to 1 \to 2 \to 4 \to 5 \to \ldots \to 26 \to 28 \to \ldots \to 49 \to 51 \to \ldots \to 63 \to 65 \to \ldots \to 81 \to 83 \to \ldots \to 99$$

Si noti che l'ordine degli elementi della seconda lista è strettamente crescente.

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;
   }
   printf("NULL\n");
   return;
}

struct nodo *costruisciLista(struct nodo *pp) {
   struct nodo *p, *q, *pq = NULL;
   int i;
   for (i=99; i>=0; i--) {
      p = pp;
      while (p != NULL && p->info != i) {
         p = p->next;
      }
      if (p == NULL) {
         q = malloc(sizeof(struct nodo));
         q->info = i;
         q->next = pq;
         pq = q;
      }
   }
   return(pq);
}


```

```
38 struct nodo *listaCasuale(int n) {
39    struct nodo *primo = NULL, *p;
40    int i;
41    srand((unsigned)time(NULL));
42    for (i=0; i<n; i++) {
43       p = malloc(sizeof(struct nodo));
44       p->info = rand() % 100;
45       p->next = primo;
46       primo = p;
47    }
48    return(primo);
49 }
50
51 int main(void) {
52    struct nodo *p, *q;
53    int n;
54    printf("Numero di elementi: ");
55    scanf("%d", &n);
56    p = listaCasuale(n);
57    stampaLista(p);
58    q = costruisciLista(p);
59    stampaLista(q);
60    return(0);
61 }
```