Corso di Informatica 1 (IN110) – Prof. Marco Liverani – a.a. 2016/2017

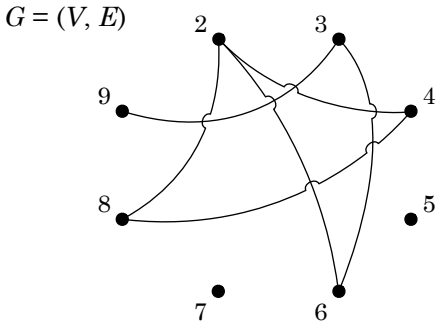
Seconda prova di esonero – 11 gennaio 2017

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.
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 intero $n > 1$, costruire le liste di adiacenza del grafo non orientato $G = (V, E)$, dove $V = \{2, 3, \ldots, n\}$ ed $E = \{(u, v) : u, v \in V \text{ e } u = kv \text{ o } v = ku \text{ per } k > 1\}$.

Esempio Sia $n = 9$. Il grafo $G = (V, E)$ è costituito dall'insieme $V = \{2, 3, 4, 5, 6, 7, 8, 9\}$ dei vertici e dall'insieme $E = \{(2, 4), (2, 6), (2, 8), (3, 6), (3, 9), (4, 8)\}$ degli spigoli; il grafo G è rappresentato in figura.



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 *V[], int n) {
  int i;
  for (i=2; i<=n; i++) {
    printf("%2d: ", i);
    stampaLista(V[i]);
  }
  return;
}

void aggiungiSpigolo(struct nodo *V[], int x, int y) {
  struct nodo *p;
  p = malloc(sizeof(struct nodo));
  p->info = y;
  p->next = V[x];
  V[x] = p;
  return;
}

void costruisciGrafo(struct nodo *V[], int n) {
  int i, j;
  for (i=2; i<=n; i++)
    V[i] = NULL;
  for (i=2; i<=n/2; i++) {
    j=2*i;
    while (j <= n) {
      aggiungiSpigolo(V, i, j);
      aggiungiSpigolo(V, j, i);
      j = j+i;
    }
  }
  return;
}

int main(void) {
  int n;
  struct nodo *V[MAX];
  printf("Inserisci n>1: ");
  scanf("%d", &n);
  costruisciGrafo(V, n);
  stampaGrafo(V, n);
  return(0);
}
```

Esercizio n. 2

Leggere in input una sequenza di numeri interi e memorizzarla in una lista L. Stampare la lista. Eliminare da L tutti gli elementi il cui valore è maggiore di tutti gli elementi che lo seguono nella lista. Stampare la lista.

Esempio Si consideri la lista

$$L = 2 \to 9 \to 7 \to 1 \to 4 \to 5 \to 2$$

Gli elementi da eliminare sono 9, 7 e 5; al termine dell'elaborazione si ottiene la lista

$$L = 2 \to \not{9} \to \not{7} \to 1 \to 4 \to \not{5} \to 2 = 2 \to 1 \to 4 \to 2$$

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;
}

struct nodo *leggiLista(void) {
  struct nodo *p, *primo=NULL;
  int n, i;
  printf("Numero di elementi della lista: ");
  scanf("%d", &n);
  printf("Elementi della lista: ");
  for (i=0; i<n; i++) {
    p = malloc(sizeof(struct nodo));
    scanf("%d", &p->info);
    p->next = primo;
    primo = p;
  }
  return(primo);
}

int daEliminare(struct nodo *p) {
  struct nodo *q=p;
  int r=0;
  while (q->next != NULL && p->info > q->next->info)
    q = q->next;
```

```
39    if (q->next == NULL && q != p)
40      r = 1;
41    return(r);
42  }
43
44  int main(void) {
45    struct nodo *p, *q, *r;
46    p = leggiLista();
47    printf("L: ");
48    stampaLista(p);
49    q = p;
50    while (q != NULL && daEliminare(q) == 1) {
51      q = q->next;
52      free(p);
53      p = q;
54    }
55    while (q != NULL && q->next != NULL) {
56      if (daEliminare(q->next) == 1) {
57        r = q->next;
58        q->next = q->next->next;
59        free(r);
60      } else {
61        q = q->next;
62      }
63    }
64    printf("L: ");
65    stampaLista(p);
66    return(0);
67  }
```