Corso di Informatica 1 (IN110) – Prof. Marco Liverani – a.a. 2019/2020

Esame scritto del 24 Gennaio 2020 (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

Letto in input un intero $n > 0$ costruire una matrice A quadrata composta da $n \times n$ numeri interi compresi tra 20 e 40, estremi inclusi. Stampare la matrice. Scambiare gli elementi delle colonne in modo tale che prima compaiano tutti gli elementi pari e dopo quelli dispari. Stampare la matrice.

Esempio Sia A la matrice random generata dal programma; dopo l'elaborazione la matrice viene trasformata, ottenendo la matrice A'.

$$A = \begin{pmatrix} 20 & 34 & 26 & 23 & 23 \\ 29 & 24 & 30 & 28 & 35 \\ 24 & 36 & 38 & 40 & 22 \\ 31 & 39 & 25 & 37 & 28 \\ 25 & 26 & 30 & 37 & 36 \end{pmatrix} \quad A' = \begin{pmatrix} 20 & 34 & 26 & 40 & 36 \\ 24 & 24 & 30 & 28 & 28 \\ 29 & 36 & 38 & 23 & 22 \\ 31 & 26 & 30 & 37 & 35 \\ 25 & 39 & 25 & 37 & 23 \end{pmatrix}$$

Soluzione

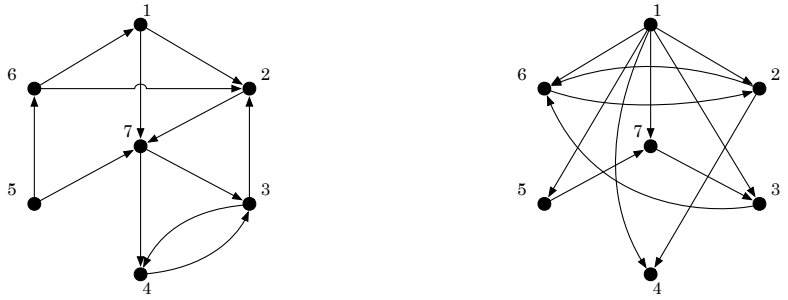
```c
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#define MAX 50

int costruisciMatrice(int A[MAX][MAX]) {
  int i, j, n;
  srand((unsigned)time(NULL));
  printf("Ordine della matrice: ");
  scanf("%d", &n);
  for (i=0; i<n; i++)
    for (j=0; j<n; j++)
      A[i][j] = rand() % 21 + 20;
  return(n);
}

void stampaMatrice(int A[MAX][MAX], int n) {
  int i, j;
  for (i=0; i<n; i++) {
    for (j=0; j<n; j++)
      printf("%2d ", A[i][j]);
    printf("\n");
  }
```

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

void scambia(int *a, int *b) {
  int c;
  c = *a;
  *a = *b;
  *b = c;
  return;
}

void modificaMatrice(int A[MAX][MAX], int n) {
  int i, j, k;
  for (j=0; j<n; j++) {
    i = 0;
    k = n-1;
    while (i<k) {
      while (i<k && A[i][j]%2 == 0)
        i++;
      while (i<k && A[k][j]%2 == 1)
        k--;
      if (i<k)
        scambia(&A[i][j], &A[k][j]);
    }
  }
  return;
}

int main(void) {
  int A[MAX][MAX], n;
  n = costruisciMatrice(A);
  stampaMatrice(A, n);
  modificaMatrice(A, n);
  stampaMatrice(A, n);
  return(0);
}
```

Esercizio n. 2

Leggere in input le liste di adiacenza di un grafo orientato $G = (V, E)$, con $V = \{1, 2, \ldots, n\}$; stampare le liste di adiacenza di G. Eliminare tutti gli spigoli (i, j) tali che i è pari e j è dispari, o viceversa, e aggiungere, per ogni $i = 1, 2, \ldots, n$, gli spigoli (i, j) tali che j è multiplo di i. Stampare le liste di adiacenza del grafo G modificato.

Esempio Si consideri il grafo $G = (V, E)$ con $n = 7$ vertici rappresentato in figura sulla sinistra. Al termine dell'elaborazione il grafo corrisponde al grafo G' rappresentato sulla destra.



Soluzione

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

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

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

int leggiGrafo(struct nodo *G[]) {
  int i, n;
  printf("Numero di vertici: ");
  scanf("%d", &n);
  for (i=1; i<=n; i++) {
    printf("Lista di adiacenza di %d\n", i);
    G[i] = leggiLista();
  }
```

```
32    return(n);
33  }
34
35  void stampaLista(struct nodo *p) {
36    while (p != NULL) {
37      printf("%d --> ", p->info);
38      p = p->next;
39    }
40    printf("NULL\n");
41    return;
42  }
43
44  void stampaGrafo(struct nodo *G[], int n) {
45    int i;
46    for (i=1; i<=n; i++) {
47      printf("%d: ", i);
48      stampaLista(G[i]);
49    }
50    return;
51  }
52
53  void eliminaSpigolo(struct nodo *G[], int i, int j) {
54    struct nodo *p, *prec;
55    p = G[i];
56    prec = NULL;
57    while (p != NULL) {
58      if (p->info == j) {
59        if (prec == NULL) {
60          G[i] = p->next;
61          free(p);
62          p = G[i];
63        } else {
64          prec->next = p->next;
65          free(p);
66          p = prec->next;
67        }
68      } else {
69        prec = p;
70        p = p->next;
71      }
72    }
73    return;
74  }
75
76  int adiacente(struct nodo *G[], int i, int j) {
77    struct nodo *p;
78    int rc = 0;
79    p = G[i];
80    while (p != NULL && p->info != j)
81      p = p->next;
82    if (p != NULL)
83      rc = 1;
84    return(rc);
85  }
86
87  void aggiungiSpigolo(struct nodo *G[], int i, int j) {
```

```
88    struct nodo *p;
89    p = malloc(sizeof(struct nodo));
90    p->info = j;
91    p->next = G[i];
92    G[i] = p;
93    return;
94  }
95
96  void modificaGrafo(struct nodo *G[], int n) {
97    int i, j;
98    struct nodo *p, *prec;
99    for (i=1; i<=n; i++) {
100     p = G[i];
101     prec = NULL;
102     while (p != NULL) {
103       if ((p->info % 2 == 0 && i % 2 == 1) || (p->info % 2 == 1 && i % 2 == 0))
104         eliminaSpigolo(G, i, p->info);
105       p = p->next;
106     }
107     for (j=2*i; j<=n; j=j+i) {
108       if (!adiacente(G, i, j))
109         aggiungiSpigolo(G, i, j);
110     }
111   }
112   return;
113 }
114
115 int main(void) {
116   int n;
117   struct nodo *G[MAX];
118   n = leggiGrafo(G);
119   stampaGrafo(G, n);
120   modificaGrafo(G, n);
121   stampaGrafo(G, n);
122   return(0);
123 }
```