Corso di Algoritmi e Strutture Dati (IN110) – Prof. Marco Liverani – a.a. 2025/2026

Seconda prova di esonero – 14 gennaio 2026

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; non è consentito l'uso di strumenti digitali (computer, tablet, smartphone, ecc.). I compiti che presenteranno evidenti ed anomale "similitudini" saranno annullati.

Esercizio n. 1

Letto in input un intero $n > 0$ generare una lista di numeri interi casuali compresi in $\{17, \ldots, 42\}$. Visualizzare in output la lista. Eliminare dalla lista tutti gli elementi ripetuti, quindi visualizzare la lista modificata.

Esempio Sia $n = 11$ e sia $L = 22 \to 31 \to 20 \to 22 \to 30 \to 31 \to 42 \to 17 \to 16 \to 31 \to 17 \to \text{null}$ una lista di numeri casuali prodotta dal programma. La lista viene così modificata: $L = 22 \to 31 \to 20 \to 30 \to 42 \to 17 \to 16 \to \text{null}$

Soluzione

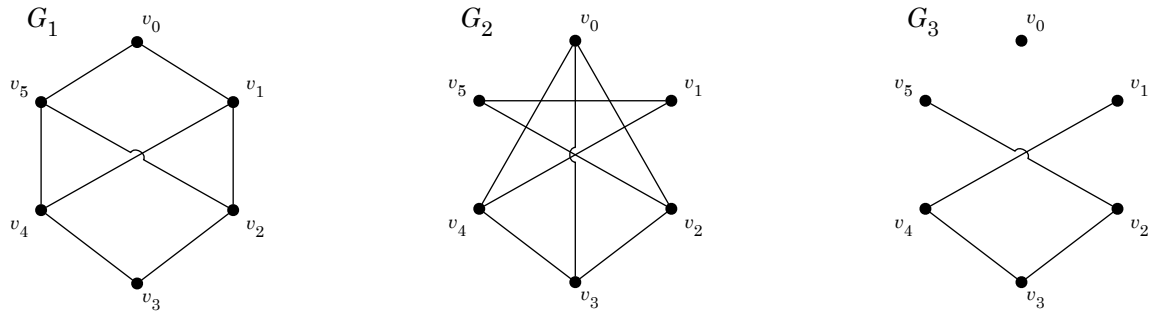
```c
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#define MAX 100

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;
}
```

```c

struct nodo *listaCasuale(void) {
  struct nodo *p, *primo = NULL;
  int n, i;
  srand((unsigned)time(NULL));
  printf("Numero di elementi: ");
  scanf("%d", &n);
  for (i = 0; i<n; i++) {
    p = malloc(sizeof(struct nodo));
    p->info = rand() % (42-17+1) + 17;
    p->next = primo;
    primo = p;
  }
  return primo;
}

void modificaLista(struct nodo *primo) {
  struct nodo *p, *q, *r;
  p = primo;
  while (p->next != NULL) {
    q = p->next;
    r = p;
    while (q != NULL) {
      if (q->info == p->info) {
        r->next = q->next;
        free(q);
        q = r->next;
      } else {
        r = q;
        q = q->next;
      }
    }
    p = p->next;
  }
  return;
}

int main(void) {
  struct nodo *p;
  p = listaCasuale();
  stampaLista(p);
  modificaLista(p);
  stampaLista(p);
  return 0;
}
```

Esercizio n. 2

Lette in input le liste di adiacenza di due grafi non orientati con n vertici, $G_1 = (V, E_1)$ e $G_2 = (V, E_2)$, costruire le liste di adiacenza del grafo $G_3 = (V, E_1 \cap E_2)$. Stampare le liste di adiacenza di G_3.

Esempio Si considerino i grafi G_1, G_2 e G_3 rappresentati in figura. Gli spigoli in comune tra i grafi G_1 e G_2 sono $E_1 \cap E_2 = \{(v_1, v_4), (v_2, v_5), (v_2, v_3), (v_3, v_4)\}$. Di conseguenza il grafo G_3 prodotto dal programma è quello 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 *G[], int n) {
  int i;
  for (i=0; i<n; i++) {
    printf("%2d: ", i);
    stampaLista(G[i]);
```

```
24    }
25    return;
26  }
27
28  struct nodo *leggiLista() {
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 adiacente(struct nodo *G[], int u, int v) {
55    int r = 0;
56    struct nodo *p;
57    p = G[u];
58    while (p != NULL && p->info != v) {
59      p = p->next;
60    }
61    if (p != NULL)
62      r = 1;
63    return r;
64  }
65
66  void aggiungiSpigolo(struct nodo *G[], int u, int v) {
67    struct nodo *p;
68    p = malloc(sizeof(struct nodo));
69    p->info = v;
70    p->next = G[u];
```

```
71    G[u] = p;
72    return;
73  }
74
75  int main(void) {
76    struct nodo *G1[MAX], *G2[MAX], *G3[MAX], *p;;
77    int i, n;
78    printf("\nGrafo G1:\n");
79    n = leggiGrafo(G1);
80    printf("\nGrafo G2:\n");
81    n = leggiGrafo(G2);
82    for (i=0; i<n; i++)
83      G3[i] = NULL;
84    for (i=0; i<n; i++) {
85      p = G1[i];
86      while (p != NULL) {
87        if (adiacente(G2, i, p->info))
88          aggiungiSpigolo(G3, i, p->info);
89        p = p->next;
90      }
91    }
92    printf("\nGrafo G3:\n");
93    stampaGrafo(G3, n);
94    return 0;
95  }
```