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

Seconda prova di esonero – 16 gennaio 2020

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 $n > 0$, costruire una lista A di n interi casuali compresi tra 0 e 99 (estremi inclusi). Stampare la lista. Costruire una seconda lista B in cui l'elemento i-esimo è dato dalla somma dei primi i elementi pari della lista A.

Esempio Sia $n = 10$. Sia A la seguente lista di numeri casuali:

$$A = 13 \to 6 \to 8 \to 5 \to 7 \to 4 \to 27 \to 44 \to 2 \to 75$$

Allora la lista B è la seguente:

$$B = 6 \to 14 \to 18 \to 62 \to 64$$

Soluzione

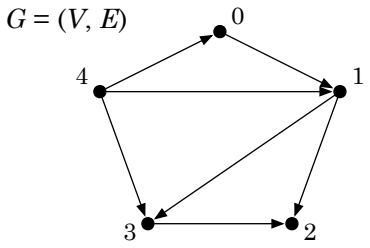
```c
#include <stdlib.h>
#include <stdio.h>
#include <time.h>

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

struct nodo *generaLista(void) {
  int i, n;
  struct nodo *p, *primo=NULL;
  printf("Numero di elementi: ");
  scanf("%d", &n);
  srand((unsigned)time(NULL));
  for (i=0; i<n; i++) {
    p = malloc(sizeof(struct nodo));
    p->info = rand() % 100;
    p->next = primo;
    primo = p;
  }
```

```
  return(primo);
}

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

struct nodo *secondaLista(struct nodo *p) {
  struct nodo *q, *primo=NULL, *ultimo=NULL;
  while (p != NULL) {
    q = malloc(sizeof(struct nodo));
    while (p != NULL && p->info %2 != 0) {
      p = p->next;
    }
    if (p != NULL) {
      if (ultimo != NULL) {
        q->info = ultimo->info + p->info;
        q->next = NULL;
        ultimo->next = q;
        ultimo = q;
      } else {
        q->info = p->info;
        q->next = NULL;
        primo = q;
        ultimo = q;
      }
      p = p->next;
    }
  }
  return(primo);
}

int main(void) {
  struct nodo *p1, *p2;
  p1 = generaLista();
  p2 = secondaLista(p1);
  stampaLista(p1);
  stampaLista(p2);
  return(0);
}
```

Esercizio n. 2

Lette in input le liste di adiacenza di un grafo $G = (V, E)$ orientato con n vertici, stampare il numero di vertici di grado entrante i per $i = 0, 1, \ldots, n-1$.

Esempio Si consideri il grafo $G = (V, E)$ con $n = 5$ vertici rappresentato in figura. G ha 1 vertice di grado entrante 0, 1 vertice di grado entrante 1, 3 vertici di grado entrante 2 e nessun vertice di grado entrante 4.



Soluzione

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

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

struct nodo *leggiLista(void) {
  struct nodo *p, *primo=NULL;
  int i, n;
  printf("Numero di elementi: ");
  scanf("%d", &n);
  printf("Inserisci %d elementi: ", 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=0; i<n; i++) {
    printf("Lista di adiacenza del vertice %d: ", i);
    G[i] = leggiLista();
```

```
  }
  return(n);
}

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]);
  }
  return;
}

void gradoVertici(struct nodo *G[], int n) {
  int A[MAX], B[MAX], i;
  struct nodo *p;
  for (i=0; i<n; i++) {
    A[i] = 0;
    B[i] = 0;
  }
  for (i=0; i<n; i++) {
    p = G[i];
    while (p != NULL) {
      A[p->info]++;
      p = p->next;
    }
  }
  for (i=0; i<n; i++)
    B[A[i]]++;
  for (i=0; i<n; i++)
    printf("%d vertici di grado entrante %d\n", B[i], i);
  return;
}

int main(void) {
  struct nodo *G[MAX];
  int n;
  n = leggiGrafo(G);
  stampaGrafo(G, n);
  gradoVertici(G, n);
  return(0);
}
```