Corso di Informatica 1 (IN110) – Prof. Marco Liverani – a.a. 2010/2011

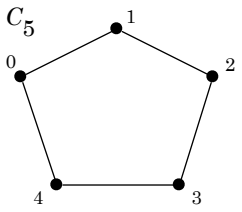
Seconda prova di esonero – 14 gennaio 2011

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 intero $n > 2$ costruire le liste di adiacenza del grafo non orientato C_n (il ciclo con n vertici). Stampare le liste di adiacenza del grafo.

Esempio Sia $n = 5$; il grafo C_5 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;
}


```

```c
void stampaGrafo(struct nodo *G[], int n) {
  int i;
  for (i=0; i<n; i++) {
    printf("G[%d]: ", i);
    stampaLista(G[i]);
  }
  return;
}

void costruisciCiclo(struct nodo *G[], int n) {
  int i;
  struct nodo *p;
  for (i=0; i<n; i++) {
    p = malloc(sizeof(struct nodo));
    p->info = i+1;
    p->next = NULL;
    G[i] = p;
    p = malloc(sizeof(struct nodo));
    p->info = i-1;
    p->next = G[i];
    G[i] = p;
  }
  G[0]->info = n-1;
  G[n-1]->next->info = 0;
  return;
}

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

Esercizio n. 2

Letto in input il numero intero $n > 0$, costruire una lista di n numeri casuali $a_1, a_2, \ldots, a_n$ maggiori di 0 e minori di 10. Stampare la lista. Costruire una seconda lista in cui ogni elemento a_i della prima lista sia ripetuto a_i volte, mantenendo lo stesso ordine degli elementi della prima lista. Stampare la seconda lista.

Esempio Sia $n = 5$ e supponiamo che la prima lista, costituita da n numeri casuali maggiori di 0 e minori di 10 sia la seguente:

$$3 \to 2 \to 3 \to 1 \to 4$$

Allora la seconda lista costruita dal programma sarà la seguente:

$$\underbrace{3 \to 3 \to 3}_{\text{3 elementi}} \to \underbrace{2 \to 2}_{\text{2 elementi}} \to \underbrace{3 \to 3 \to 3}_{\text{3 elementi}} \to \underbrace{1}_{\text{1 elem.}} \to \underbrace{4 \to 4 \to 4 \to 4}_{\text{4 elementi}}$$

Si noti che l'ordine degli elementi della seconda lista, riflette l'ordine degli elementi della prima.

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 *p) {
  struct nodo *q, *primo = NULL, *ultimo = NULL;
  int i;
  while (p != NULL) {
    for (i=0; i<p->info; i++) {
      q = malloc(sizeof(struct nodo));
      q->info = p->info;
      q->next = NULL;
      if (primo == NULL)
        primo = q;
      if (ultimo != NULL)
        ultimo->next = q;
      ultimo = q;
    }
    p = p->next;
  }
  return(primo);
```

```
}

struct nodo *listaCasuale(int n) {
  struct nodo *primo = NULL, *p;
  int i;
  srand((unsigned)time(NULL));
  for (i=0; i<n; i++) {
    p = malloc(sizeof(struct nodo));
    p->info = rand() % 9 + 1;
    p->next = primo;
    primo = p;
  }
  return(primo);
}

int main(void) {
  struct nodo *p, *q;
  int n;
  printf("Numero di elementi: ");
  scanf("%d", &n);
  p = listaCasuale(n);
  stampaLista(p);
  q = costruisciLista(p);
  stampaLista(q);
  return(0);
}
```