LISTE

INSERIMENTO ORDINATO

```
list sortInsertRic(list l, int content) {
    node *n;
    if (l != NULL && l->content < content) {
        l->next = sortInsert(l->next, content);
        return l;
    } else {
        n = (node *) malloc(sizeof(node));
        n->content = content;
        if (l != NULL) {
            n->next = l;
        } else {
            n->next = NULL;
        }
        return n;
    }
}

list sortInsertIte(list l, int content) {
  list currentNode = l;
  node *n = (node *) malloc(sizeof(node));

  // Casi in cui l'elemento deve essere inserito in prima posizione
  if (l == NULL || l->content >= content) {
    return headInsert(l, content);
  }

  // Caso in cui l'elemento deve essere inserito in una posizione centrale
  while (currentNode->next != NULL && currentNode->next->content < content) {
    currentNode = currentNode->next;
  }
  n->next = currentNode->next;
  n->content = content;
  currentNode->next = n;
  return l;
}
```

1. Scrivere un programma che, attraverso una lista, gestisca una pila di caratteri. La pila è una lista nella quale i nuovi elementi vengono sempre inseriti in testa (push). Se si vuole leggere un elemento della lista si compie l'opzione pop, che estrae dalla testa l'elemento.

Il secondo metodo serve per specificare come va usata una lista per la quale si deve modificare la testa senza la return

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

typedef struct nodo{
    char carattere;
    struct nodo* next;
} nodo;

typedef nodo* ptrNodo;

ptrNodo inserisciInPila(ptrNodo lista);
ptrNodo rimuoviDaPila(ptrNodo lista, char *letto);
char rimuoviDaPila2(ptrNodo *lista);
void stampaLista(ptrNodo lista);

int main(int argc, const char * argv[]) {
    ptrNodo lista = NULL;
    int c;
    char cc;

    do{
        printf("\nMENU\n");
        printf("1) Inserici nella pila\n");
        printf("2) Rimuovi dalla pila\n");
        printf("3) Stampa la pila\n\n");
        printf(">> ");
        scanf("%d",&c);

        switch(c){
            case 1:
                lista = inserisciInPila(lista);
                break;
            case 2:
                if (lista == NULL)
                    printf("La lista è vuota\n");
                else{
                    //lista = rimuoviDaPila(lista, &cc);
                    //printf("Il carattere letto è: %c\n",cc);
                    printf("Il carattere letto è:
%c",rimuoviDaPila2(&lista));
                }
                break;
            case 3:
                stampaLista(lista);
                break;
        }
    }while (c!=0);
    return 0;
```

```
}

ptrNodo inserisciInPila(ptrNodo lista)
{
    ptrNodo nodo;
    nodo = (ptrNodo)malloc(sizeof(nodo));
    printf("Inserisci carattere: ");
    fpurge(stdin);
    scanf("%c",&(nodo->carattere));
    if (lista == NULL)
        nodo->next = NULL;
    else
        nodo->next = lista;
    return nodo;
}

ptrNodo rimuoviDaPila(ptrNodo lista, char *letto){
    ptrNodo temp;

    if (lista == NULL)
    {
        *letto = NULL;
        return NULL;
    }
    else{
        temp = lista;
        lista = lista->next;
        *letto = temp->carattere;
        free(temp);
        return lista;
    }
}

char rimuoviDaPila2(ptrNodo *lista)
{
    char c;
    ptrNodo temp;
    if (*lista == NULL)
        return NULL;
    else
    {
        temp = *lista;
        *lista = (*lista)->next;
        c = temp->carattere;
        free (temp);
        return c;
    }
}

void stampaLista(ptrNodo lista)
{
    if (lista == NULL)
        return;
    printf("%c ", lista->carattere);
    stampaLista(lista->next);
}
```

2. Scrivere un programma che, attraverso una lista, gestisca una coda di persone. Ogni persona è identificata da un numero. La coda riceve nuove persone dalla coda, e le elimina dalla testa. Scrivere le funzioni che permettono di inserire e leggere/rimuovere le persone dalla coda.*/

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

typedef struct persona{
    int id_persona;
    char nome[20];
    char cognome[20];
    struct persona *next;
} persona;

typedef persona *ptrPersona;

ptrPersona inserisciInCoda(ptrPersona lista);
ptrPersona inserisciInCodaRic(ptrPersona lista);
int rimuoviDallaCoda(ptrPersona *lista);
void stampaLista(ptrPersona lista);

int main(int argc, const char * argv[]) {
    ptrPersona lista = NULL;
    int c;

    do{
        printf("\nMENU\n");
        printf("1) Inserisci in coda\n");
        printf("2) Prossimo cliente\n");
        printf("3) Stampa lista clienti\n");
        printf(">> ");
        scanf("%d",&c);

        switch(c){
            case 1:
                lista = inserisciInCoda(lista);
                //lista = inserisciInCodaRic(lista);
                break;
            case 2:
                printf("Il prossimo cliente è:
%d",rimuoviDallaCoda(&lista));
                break;
            case 3:
                stampaLista(lista);
                break;
        }
    }while(c!=0);
    return 0;
}

ptrPersona inserisciInCoda(ptrPersona lista){
    ptrPersona nodo;
    ptrPersona testa = lista;

    nodo = (ptrPersona)malloc(sizeof(persona));
```

```
    printf("Inserisci ID cliente: ");
    scanf("%d",&nodo->id_persona);
    printf("Inserisci nome: ");
    scanf(" %s",nodo->nome);
    printf("Inserisci cognome: ");
    scanf(" %s",nodo->cognome);
    nodo->next = NULL;

    if (lista==NULL)
    {
        return nodo;
    }
    while(lista->next !=NULL)
        lista = lista->next;
    lista->next = nodo;
    return testa;

}

ptrPersona inserisciInCodaRic(ptrPersona lista){
    if (lista == NULL)
    {
        lista = (ptrPersona)malloc(sizeof(persona));
        printf("Inserisci ID cliente: ");
        scanf("%d",&lista->id_persona);
        printf("Inserisci nome: ");
        scanf(" %s",lista->nome);
        printf("Inserisci cognome: ");
        scanf(" %s",lista->cognome);
        lista->next = NULL;
        return lista;
    }

    lista->next = inserisciInCodaRic(lista->next);
    return lista;

}

int rimuoviDallaCoda(ptrPersona *lista){
    ptrPersona temp;
    int i;

    if ((*lista) == NULL)
        return NULL;
    else{
        temp = (*lista);
        i = temp->id_persona;
        (*lista) = (*lista)->next;
        free (temp);
        return i;
    }

}

void stampaLista(ptrPersona lista){
    if (lista==NULL)
```

```
        return;
    printf("ID Cliente: %d\n",lista->id_persona);
    printf("Nome Cliente: %d\n",lista->nome);
    printf("Cognome Cliente: %d\n\n",lista->cognome);
    stampaLista(lista->next);
}
```

3. Scrivere un programma che, attraverso una lista, permette di gestire le luci dell'albero di natale.
   La lista contiente:
   - Il numero del led;
   - Il colore
   - la posizione X
   - la posizione Y
   - Acceso o spento.

   Il programma permette di:
   - Collegare un nuovo led in ordine crescente
   - Scollegare un led dalla coda
   - Switchare led accesi con led spenti
   - Creare una lista contenente solo i led rossi;

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

typedef struct nodo{
    int nLed;
    char colore;
    int x;
    int y;
    char stato[4];
    struct nodo *next;
} nodo;

typedef nodo *ptrNodo;

ptrNodo collegaLed(ptrNodo lista, int nLed);
ptrNodo scollegaLed(ptrNodo lista);
ptrNodo creaLista(ptrNodo lista);
ptrNodo addLedToList(ptrNodo lista, int nLed, char colore, int x, int y,
char stato[]);
void  switchOnOff(ptrNodo albero);
void stampaLed(ptrNodo lista);

int main(int argc, const char * argv[]) {
    ptrNodo albero = NULL;
    ptrNodo ledRossi = NULL;
    int c, nLed;

    do{
        printf("\nMenu\n");
        printf("1) Collega led\n");
        printf("2) Scollega led in fondo\n");
        printf("3) Switch ON/OFF\n");
        printf("4) Crea lista led rossi\n");
        printf("5) Stampa\n\n");
        printf(">> ");
        scanf("%d",&c);
        switch(c){
            case 1:
                printf("Inserisci il numero del led: ");
                scanf("%d",&nLed);
```

```
                albero = collegaLed(albero, nLed);
                break;
            case 2:
                albero = scollegaLed(albero);
                break;
            case 3:
                switchOnOff(albero);
                break;
            case 4:
                ledRossi = creaLista(albero);
                break;
            case 5:
                stampaLed(albero);
                printf("\n\nLED ROSSI:\n");
                stampaLed(ledRossi);
                break;
        }
    }while(c!=0);
    return 0;
}

ptrNodo collegaLed(ptrNodo lista, int nLed){
    ptrNodo led;
    if (lista != NULL && lista->nLed < nLed){
        lista->next = collegaLed(lista->next, nLed);
        return lista;
    }
    else{
        led = (ptrNodo)malloc(sizeof(nodo));
        led->nLed = nLed;
        printf("Inserisci colore: ");
        fpurge(stdin);
        scanf("%c",&led->colore);
        printf("Inserisci x: ");
        scanf("%d",&led->x);
        printf("Inserisci y: ");
        scanf("%d",&led->y);
        printf("Inserisci stato: ");
        fpurge(stdin);
        scanf("%s",led->stato);
        if (lista==NULL)
            led->next = NULL;
        else
            led->next = lista;
        return led;
    }
}

ptrNodo scollegaLed(ptrNodo lista){
    ptrNodo temp;
    if (lista == NULL)
        return NULL;
    if (lista->next==NULL)
    {
        free(lista);
        return NULL;
```

```
    }
    if (lista->next->next==NULL)
    {
        temp = lista->next;
        lista->next = NULL;
        free(temp);
        return lista;
    }
    else
    {
         lista->next = scollegaLed(lista->next);
        return lista;
    }

}

ptrNodo creaLista(ptrNodo lista){
    ptrNodo newLista = NULL;
    if (lista==NULL)
        return NULL;
    else{
        if ((lista->colore=='R') || (lista->colore=='r')){
            newLista = addLedToList(newLista, lista->nLed, lista-
>colore, lista->x, lista->y, lista->stato);
            newLista->next = creaLista(lista->next);
        }
        else
            newLista = creaLista(lista->next);
        return newLista;
    }

}

void  switchOnOff(ptrNodo albero)
{
    if (albero==NULL)
        return;
    else{
        if (strcmp(albero->stato,"ON")==0)
            strcpy(albero->stato,"OFF");
        else if (strcmp(albero->stato,"OFF")==0)
            strcpy(albero->stato,"ON");
        switchOnOff(albero->next);
        return;
    }
}

void stampaLed(ptrNodo lista){
    if (lista==NULL)
        return;
    else
    {
        printf("LED: %d\nCOLORE: %c\nX: %d Y: %d\nSTATO: %s\n",lista-
>nLed,lista->colore,lista->x,lista->y,lista->stato);
        stampaLed(lista->next);
    }
```

```
}

ptrNodo addLedToList(ptrNodo lista, int nLed, char colore, int x, int y,
char stato[]){
    ptrNodo led;
    if (lista != NULL && lista->nLed < nLed){
        lista->next = addLedToList(lista->next, nLed, colore, x, y,
stato);
        return lista;
    }
    else{
        led = (ptrNodo)malloc(sizeof(nodo));
        led->nLed = nLed;
        led->colore = colore;
        led->x = x;
        led->y = y;
        strcpy(led->stato,stato);
        if (lista==NULL)
            led->next = NULL;
        else
            led->next = lista;
        return led;
    }
}
```