```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define  Max = 20

typedef struct studente{
    char nome[20];
    char cognome[20];
    int data;
    double media;
    struct studente * next;
}stud_t;

typedef stud_t * ptrstud;

ptrstud inserisciInTesta (ptrstud studente);
ptrstud inserisciInCoda (ptrstud studente);
void ordinaLista (ptrstud studente, ptrstud testa , int k);
ptrstud inserisciOrdinato (ptrstud studente, ptrstud nodo);
ptrstud inserisciInPosizione (ptrstud studente, int count);
ptrstud listaNati1992 (ptrstud studente, ptrstud lista);
void stampaNati1992 (ptrstud studente);
void stampalista ( ptrstud studente);
void ordinaListaVoti( ptrstud studente, ptrstud testa, int k);

int main() {
    int s = 0;
   int pos = 0;
    ptrstud studente = NULL;
    ptrstud testa;
    ptrstud nodo;
    ptrstud lista = NULL;


    do{
        printf("Menù:\n");
        printf("1) Inserisci studente in testa\n");
        printf("2) Inserisci studente in coda\n");
        printf("3) Per una sorpresa\n");
        printf("4) Inserisci studente in ordine alfabetico\n");
        printf("5) Ordina lista\n");
```

```
        printf("6) Inserisci in posizione scelta dall'utente\n");
        printf("7) Stampa la lista\n");
        printf("8) Stampa la lista in modo ordinato\n");
        printf("9) Stampa solo i nati nel 1992\n");
        printf("10) Esci\n");
        printf(">> ");

        scanf("%d", &s);

        switch (s){
        case 1 :
           studente = inserisciInTesta(studente);
              break;
        case 2:
              studente = inserisciInCoda(studente);
              break;
        case 3:
              printf(" <3 un cuoricino per te... Niente avevo saltato un numero nel menù e mi seccava
cancellare e riscrivere tutto, quindi ho solo aggiunto un nuovo punto :)");
              break;
        case 4:
              testa = studente;
              ordinaLista(studente, testa, 0);
              stampalista(studente);

              nodo = (ptrstud)malloc(sizeof(stud_t));
              if (nodo) {
                 printf("Nome studente: ");
                 scanf(" %s", nodo->nome);
                 printf("\nCognome studente: ");
                 scanf(" %s", nodo->cognome);
                 printf("\nData di nascita: ");
                 scanf(" %d", &(nodo->data));
                 printf("\nMedia voti: ");
                 scanf(" %lf", &(nodo->media));
              } else
                 printf("Errore allocazione memoria\n");

              studente = inserisciOrdinato(studente, nodo);

           break;
```

```
        case 5:
                testa = studente;
                ordinaLista( studente, testa, 0);
            break;
        case 6:
                printf("In che posizione si desidera inserire il nuovo studente?  ");
                scanf(" %d", &pos);
                studente = inserisciInPosizione(studente, pos);
            break;
        case 7:
            stampalista(studente);
                break;
        case 8:
                printf("mi ruppi u cazz\n");
               // stampaOrdinato(studente);
             break;
        case 9:
                stampaNati1992(studente);

                //stampalista(listaNati1992(studente, lista));
             break;
        case 10:
                testa = studente;
                 ordinaListaVoti(  studente,  testa,  0);
        }
} while (s < 11);
    return 0;
}

ptrstud inserisciInTesta (ptrstud studente){
    ptrstud nodo;
    nodo = (ptrstud)malloc(sizeof(stud_t));
    printf("Nome studente: ");
    scanf(" %s", nodo->nome);
    printf("\nCognome studente: ");
    scanf(" %s", nodo->cognome);
    printf("\nData di nascita: ");
    scanf(" %d", &(nodo->data));
    printf("\nMedia voti: ");
    scanf(" %lf", &(nodo->media));
```

```
    nodo->next = studente;
    return nodo;
}



ptrstud inserisciInCoda (ptrstud studente){
    ptrstud nodo = NULL;

    if (studente == NULL){
        nodo = (ptrstud)malloc(sizeof(stud_t));
        printf("Nome studente: ");
        scanf(" %s", nodo->nome);
        printf("\nCognome studente: ");
        scanf(" %s", nodo->cognome);
        printf("\nData di nascita: ");
        scanf(" %d", &(nodo->data));
        printf("\nMedia voti: ");
        scanf(" %lf", &(nodo->media));
        nodo->next = NULL;
        return nodo;
    }else{
        studente->next = inserisciInCoda(studente->next);
        return studente;
    }
}


void ordinaLista (ptrstud studente, ptrstud testa, int k){
    char tempn [20];
    char tempc [20];
    int tempd;
    double tempm;

    if (studente == NULL || studente->next == NULL){
        if (k==0){
            return;
        } else{
            ordinaLista(testa,testa, 0);
        }
```

```
} else {
    if ( strcmp(studente->cognome, studente->next->cognome)>0)
        // disordinato
    {
        strcpy(tempc, studente->cognome);
        strcpy(studente->cognome, studente->next->cognome);
        strcpy(studente->next->cognome, tempc);
        strcpy(tempn, studente->nome);
        strcpy(studente->nome, studente->next->nome);
        strcpy(studente->next->nome, tempn);
        tempm = studente->media;
        studente->media= studente->next->media;
        studente->next->media= tempm;
        tempd = studente->data;
        studente->data= studente->next->data;
        studente->next->data= tempd;

        ordinaLista(studente->next,testa, 1);
    } else if (strcmp(studente->cognome, studente->next->cognome)==0){
        if ( strcmp(studente->nome, studente->next->nome)>0)
        {
            strcpy(tempn, studente->nome);
            strcpy(studente->nome, studente->next->nome);
            strcpy(studente->next->nome, tempn);
            strcpy(tempc, studente->cognome);
            strcpy(studente->cognome, studente->next->cognome);
            strcpy(studente->next->cognome, tempc);
            ordinaLista(studente->next,testa, 1);
            tempm = studente->media;
            studente->media= studente->next->media;
            studente->next->media= tempm;
            tempd = studente->data;
            studente->data= studente->next->data;
            studente->next->data= tempd;
        }else {
            ordinaLista(studente->next, testa, k);
        }
    } else {
        ordinaLista(studente->next, testa, k);
    }
}
```

```
}

ptrstud inserisciOrdinato (ptrstud studente, ptrstud nodo){
   if (studente != NULL && strcmp(nodo->cognome, studente->cognome)>0) {
      studente->next = inserisciOrdinato(studente->next, nodo);
      return studente;
   } else if(studente != NULL && strcmp(nodo->cognome, studente->cognome)==0) {
      if(strcmp(nodo->nome, studente->nome) > 0) {
         studente->next = inserisciOrdinato(studente->next, nodo);
         return studente;
      } else {
         nodo->next = studente;
         return nodo;
      }
   } else {
      nodo->next = studente;
      return nodo;
   }



}

ptrstud inserisciInPosizione (ptrstud studente, int count){
   ptrstud nodo = NULL;
   ptrstud testa = studente;
   ptrstud temp = NULL;
   int i;

   nodo = (ptrstud)malloc(sizeof(stud_t));
   if (nodo) {
      printf("Nome studente: ");
      scanf(" %s", nodo->nome);
      printf("\nCognome studente: ");
      scanf(" %s", nodo->cognome);
      printf("\nData di nascita: ");
      scanf(" %d", &(nodo->data));
      printf("\nMedia voti: ");
      scanf(" %lf", &(nodo->media));
   } else
      printf("Errore allocazione memoria\n");
```

```
    if (studente == NULL && count != 1) {
        printf("La lista è vuota, lo studente è il primo ad essere inserito nella lista");
        return nodo;
    } else if (studente == NULL && (count == 0 || count == 1)){
        return nodo;
    } else {
        for(i = 1; i < count-1; i++) {
            studente = studente->next;
        }
        temp = studente->next;
        studente->next = nodo;
        nodo->next = temp;
        return testa;
    }
}



void stampalista ( ptrstud studente){
    if (studente == NULL)
        return;

    printf("Nome: %s\nCognome: %s\nData di nascita: %d\nMedia voti: %.2lf\n", studente->nome,
studente->cognome, studente->data, studente->media);
    printf("\n");
    stampalista(studente->next);

    }

//non funzionano da qua
ptrstud listaNati1992 (ptrstud studente, ptrstud lista){

    if (studente!= NULL){
        for (;studente->next== NULL; studente= studente->next) {
            if (studente->data == 1992){
                if(lista == NULL){
                    lista = studente;
                    lista->next =NULL;

                }else{
                    for (; lista->next== NULL; lista = lista->next) {
                        lista->next= studente;
```

```
                lista->next->next= NULL;
            }
        }
    }
    return lista;
}else{
    printf("La lista è vuota");
    return studente;
}
}


void stampaNati1992 (ptrstud studente) {
    ptrstud nodo;
    nodo = (ptrstud)malloc(sizeof(stud_t));


    for (; studente == NULL; studente= studente->next) {
        nodo = NULL;
        if ( studente->data == 1992){
            nodo = studente;
            nodo->next = NULL;
            stampalista(nodo);
        }
    }
    return;
}

//funziona

void ordinaListaVoti( ptrstud studente, ptrstud testa, int k){
    char tempn [20];
    char tempc [20];
    int tempd;
    double tempm;

    if (studente == NULL || studente->next == NULL){
        if (k==0){
            return;
        } else{
```

```
            ordinaLista(testa,testa, 0);
        }

    } else {
        if (studente->media < studente->next->media)
            // disordinato
        {
            tempm = studente->media;
            studente->media= studente->next->media;
            studente->next->media= tempm;
            tempd = studente->data;
            studente->data= studente->next->data;
            studente->next->data= tempd;
            strcpy(tempc, studente->cognome);
            strcpy(studente->cognome, studente->next->cognome);
            strcpy(studente->next->cognome, tempc);
            strcpy(tempn, studente->nome);
            strcpy(studente->nome, studente->next->nome);
            strcpy(studente->next->nome, tempn);

            ordinaLista(studente->next,testa, 1);

        }else {
            ordinaLista(studente->next, testa, k);
        }
    }
}
```