```
#include<iostream>

#include<cstring>

#include<cstdlib>

#include<ctime>



using namespace std;


struct Student{

int id;

char name[50];

float cgpa;

};



//Add student and expand memory safely

void addStudent(Student*&s, int &n){

Student* temp = (Student*)realloc(s,(n+1) * sizeof(Student));

if(temp == NULL){

cout<<"Memory allocation failed!\n";

return;

}

s = temp;


cout<<"Enter ID:";

cin>>s[n].id;

cout<<"Entar Name:";

cin.ignore();
cin.getline(s[n].name,50);
cout<<"Enter CGPA:";
cin>>s[n].cgpa;

n++;
}

//Show all students
void showStudent(Student* s, int n){
cout<<"\n---Student List---\n";
for(int i =0; i<n; i++){
cout<<"ID:"<<s[i].id
   <<",Name:"<<s[i].name
   <<",CGPA:"<<s[i].cgpa<<endl;
}
}

//Linear Search by ID
int linearSearch(Student*s, int n, int id){
for(int i=0;i<n;i++){
if(s[i].id==id) {
return i;
}
}
return -1;
}

//Sort by ID (Insertion sort)
```

```
void sortByID(Student* s, int n){
        for(int i=1; i<n; i++){
                Student Key = s[i];
                int j=i-1;
                while(j>=0&&s[j].id>Key.id){
                        s[j+1] = s[j];
                        j--;
                }
                s[j+1]=Key;
        }
}


//Binary search by ID (after sorting)
int binarySearch(Student*s, int n, int id){
int i = 0;
int r = n-1;
while(i<=r){
int m=(i+r)/2;
if (s[m].id==id)return m;
else if (s[m].id<id) i=m+1;
else r = m-1;
}
return-1;
}


//Bubble sort by Name
```

```
void bubbleSortName(Student*s, int n){
for(int i=0; i<n-1; i++){
for(int j=0; j<n-i-1; j++){
if(strcmp(s[j].name,s[j+1].name)>0){
swap(s[j],s[j+1]);
}
}
}
}


//Selection sort by CGPA
void selectionSortCGPA(Student*s, int n, bool asc){
for(int i=0; i<n-1; i++){
int idx = i;
for(int j=i+1; j<n; j++){
if(asc){
if(s[j].cgpa<s[idx].cgpa)idx = j;
}else{
if(s[j].cgpa>s[idx].cgpa)idx = j;
}
}
swap(s[i],s[idx]);
}
}


//Compare linear and binary search time
```

```
void compareSearch(Student*s, int n){
if(n==0){
cout<<"No student records.\n";
return;
}

int id = s[rand()%n].id;
cout<<"Searching for ID:"<<id<<endl;



clock_t t1 = clock();
linearSearch(s,n,id);
clock_t t2 =- clock();
double linTime=double(t2-t1)/CLOCKS_PER_SEC;

sortByID(s,n);
t1=clock();
binarySearch(s,n,id);
t2=clock();
double binTime=double(t2-t1)/CLOCKS_PER_SEC;



cout<<"Linear Search Time:"<<linTime<<"s\n";
cout<<"Binary Serch Time:"<<binTime<<"s\n";
}

int main(){
```

```
Student*s=NULL;

int n=0;

int choice;

srand(time(0));


do{

cout<<"\n======Student Database Menue=====\n";

cout<<"1.Add Student\n";

cout<<"2.Show Students\n";

cout<<"3.Linear Search by ID\n";

cout<<"4.Binary Search by ID (After Sorting)\n";

cout<<"5.Sort by Name\n";

cout<<"6.Sort by CGPA\n";

cout<<"7.Compare Search Time\n";

cout<<"8.Exit\n";

cout<<"Enteer your choice:\n";

cin>>choice;


switch(choice){

case 1:{

addStudent(s, n);

break;

}

case 2:{

showStudent(s, n);

break;

}

case 3:{
```

```
int id;

cout<<"Entar ID to search:";

cin>>id;

int idx= linearSearch(s, n, id);

if(idx==-1)cout<<"Student not found.\n";

else cout<<"Found:"<<s[idx].name<<",CGPA:"<<s[idx].cgpa<<endl;

break;

}

case 4:{

sortByID(s, n);

int id;

cout<<"Entar ID to search:";

cin>>id;

int idx = binarySearch(s, n, id);

if(idx==-1) cout<<"Student not found.\n:";

else cout<<"Found:"<<s[idx].name<<",CGPA:"<<s[idx].cgpa<<endl;

break;

}


case 5:{

bubbleSortName(s, n);

cout<<"Student sort by Name.\n";

break;

}

case 6:{

int order;

cout<<"Sort by CGPA: 1.Adcending 2.Descending :";

cin>>order;
```

```
selectionSortCGPA(s,n,order==1);
cout<<"Student sort by CGPA.\n";
break;
}

case 7:{
compareSearch(s, n);
break;
}
case 8:{
cout<<"Exiting program. Goodbye!\n";
break;
}
default:{
cout<<"Invalid choice. Try again.\n";
break;
}
}
} while(choice !=8);
free(s);
return 0;
}
```