Pranay Lobo 8472

Mehek Male 8474

BATCH - D

**DATA STRUCTURES MINI PROJECT REPORT**

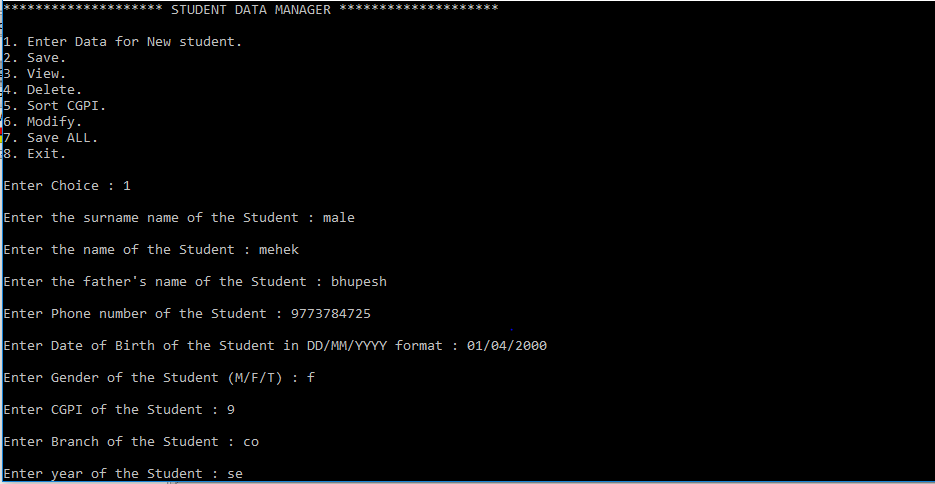
**Problem Statement:**

Designing a C Program that will work as a Student Data Manager.

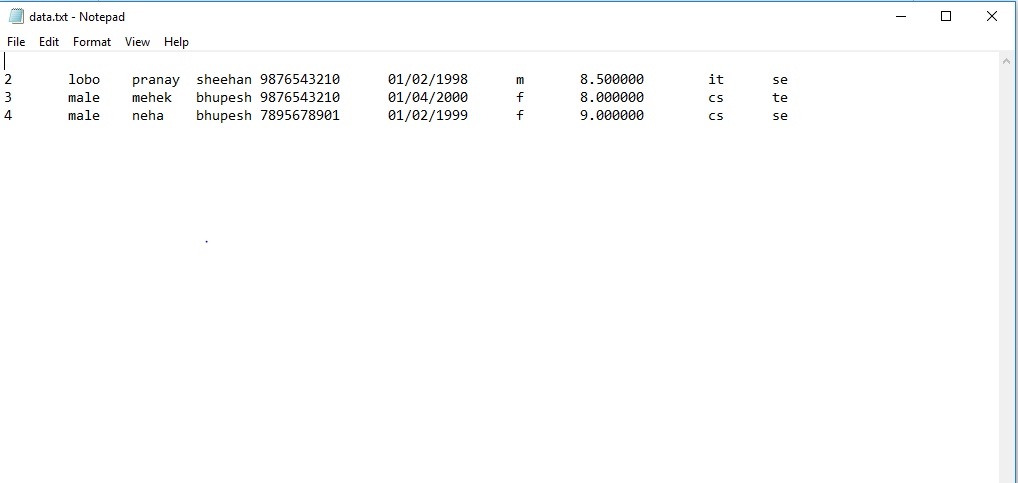
We can store the records of student In a File, and it can also be modified or removed if we want. We can sort the student according to surname in order to generate ROLL NO and also we can sort student according to their CGPI. We can search student easily using their ID or name. We can append the data of new students added and we can also sort the new data and previous data and write it in the same file.

**Functionalities supported by the project:**

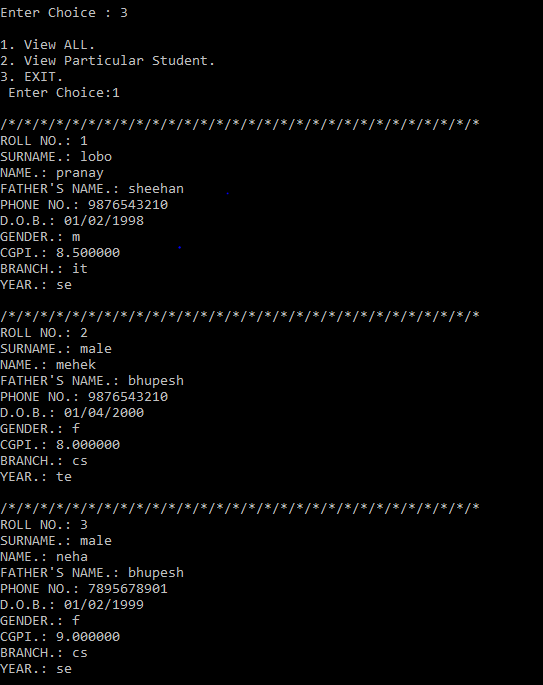
1. Add a Student record.



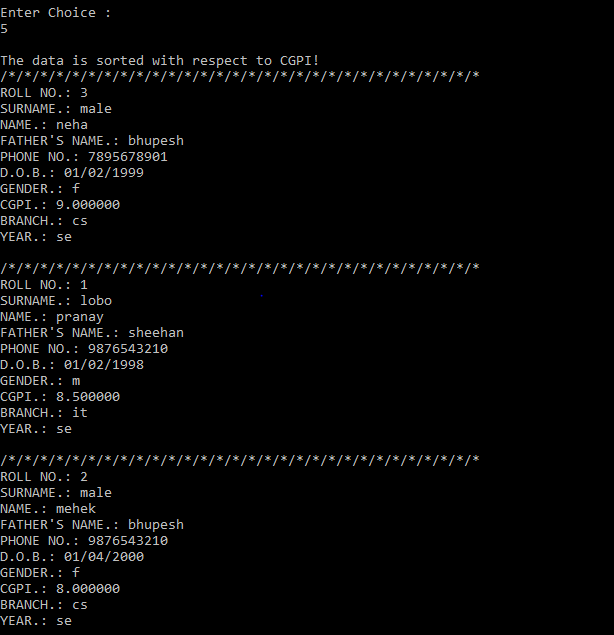
1. Save: This function will be called so that Student will be assigned a roll no according to his/her surname, this data will be appended to the existing data in the file.



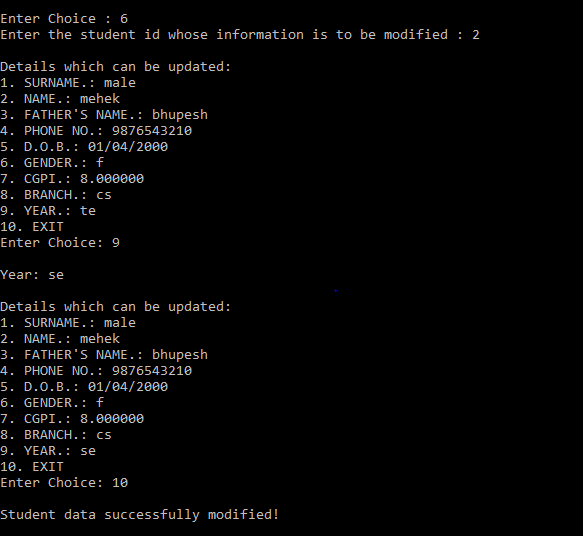
1. View Records.
2. View all Records
3. View particular Record by ID.



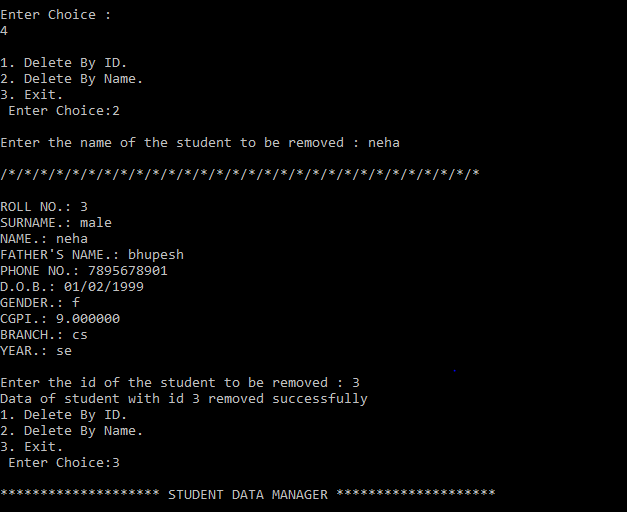
1. Sort by CGPI.

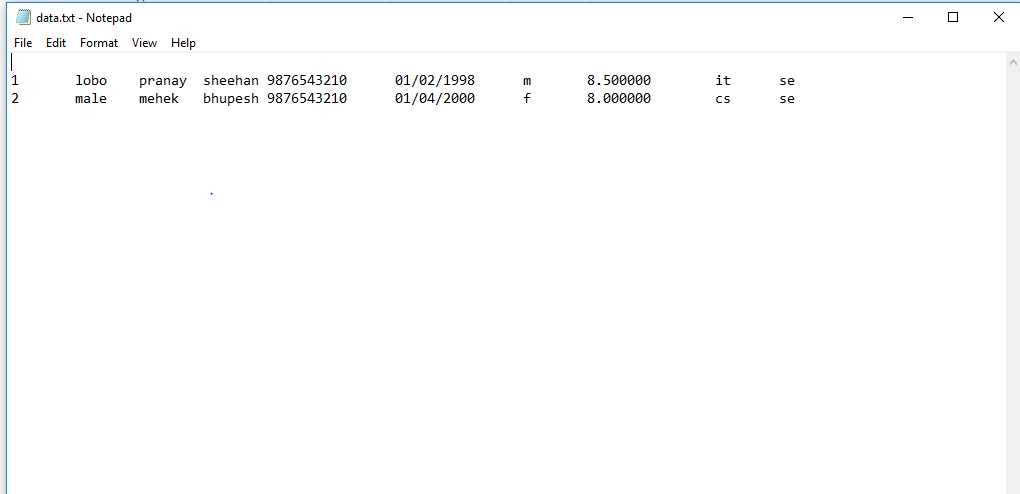


1. Modify: Using this function user will be able to modify any field in the record.

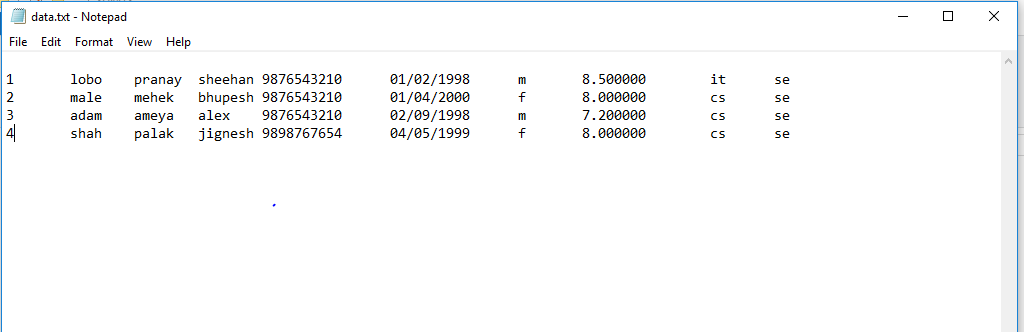


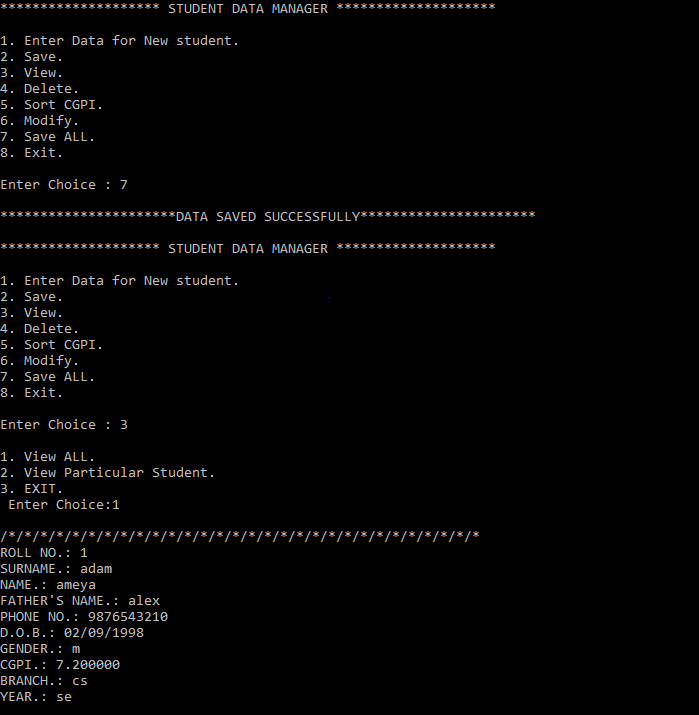
1. Remove a Student from the records.
2. Using ID.
3. Using NAME.

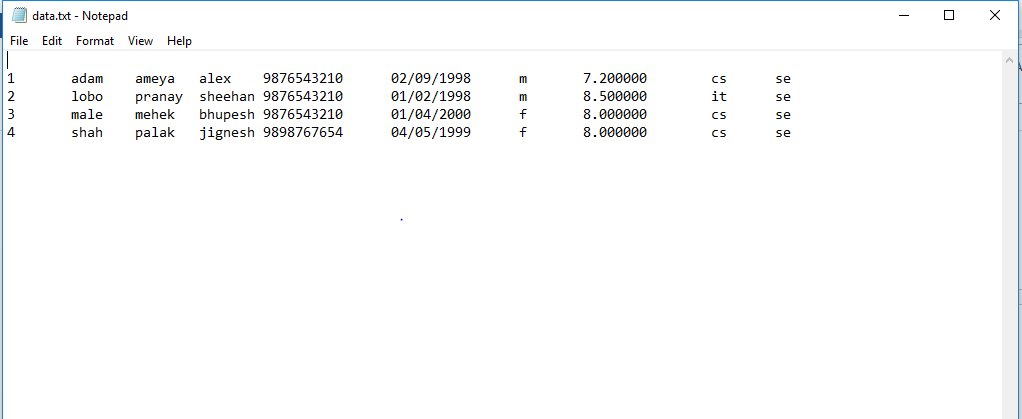




1. SaveALL: This function is called when user wants to sort the newly entered data with the previously entered data in the file. And rewrite the file as a whole containing new and previous data.

Before calling SaveALL

After calling Save ALL

File

**Selected Data Structure and reason of selecting it:**

We have used Doubly Linked List in our project, as in Doubly Linked List we have previous and next o fields using which we can traverse easily to previous and next node. Linked List also gives us advantage of memory space as the amount of data we add only that much data occupies memory.

**Algorithm:**

1. Switch case for particular operation to take place.
2. If the user selects “Enter a new Record” option then, take Students data as ”SURNAME”, “FATHER NAME”, “NAME”, “PHONE NUMBER”, “D.O.B”, “GENDER”, “CGPI”, “BRANCH” and “YEAR” and then compare SURNAME entered with previous SURNAME present in list in order to SORT and assign ID.
3. If the user selects “Delete a Record” option then, User will have a choice of deleting using ID or Delete using NAME and then we’ll search for either id or name respectively in order to delete it.
4. If the user selects “View” option then, user have to choice of whether he/she wants to view all data or view data of particular employee.
5. If the user selects “Display” option then, all Student Records will be displayed sorted according to their Roll No.
6. If the user selects “Modify” option then, user will be able to modify any field in the record.
7. Sort by CGPI: Records will be sorted according to CGPI in descending order.
8. If the user selects “EXIT” option then, Save() function will be called so that Student will be assigned a roll no according to his/her surname, this data will be appended to the existing data in the file.
9. If the user selects “SaveALL” option then, the newly entered data with the previously entered data in the file will be sorted. And rewrite the file as a whole containing new and previous data.

**Code:**

#include<stdio.h>

#include<stdlib.h>

#include<malloc.h>

#include<string.h>

void enterD();

void deleteID();

void deleteName();

void sortD();

void viewAll();

void viewP();

void sortCGPI();

void save();

void modify();

struct student

{

struct student \*prev;

int id;

char sname[20];

char name[20];

char fname[20];

char phoneno[20];

char DOB[20];

char gender[20];

float CGPI;

char branch[20];

char year[20];

struct student \*next;

};

struct student \*start=NULL;

struct student \*temp;

int version=1;

int rn=0;

int count=4;

FILE \*fptr;

void enterD()

{

char sna[20];

char na[20];

char fna[20];

char pno[20];

char dob[20];

char gen[20];

float cgpi;

char br[20];

char yr[20];

int i,n=0,nt;

struct student \*stud;

stud=(struct student \*)malloc(sizeof(struct student));

printf("\nEnter the surname name of the Student : ");

scanf("%s",&sna);

printf("\nEnter the name of the Student : ");

scanf("%s",&na);

printf("\nEnter the father's name of the Student : ");

scanf("%s",&fna);

printf("\nEnter Phone number of the Student : ");

scanf("%s",&pno);

printf("\nEnter Date of Birth of the Student in DD/MM/YYYY format : ");

scanf("%s",&dob);

printf("\nEnter Gender of the Student (M/F/T) : ");

scanf("%s",&gen);

printf("\nEnter CGPI of the Student : ");

scanf("%f",&cgpi);

printf("\nEnter Branch of the Student : ");

scanf("%s",&br);

printf("\nEnter year of the Student : ");

scanf("%s",&yr);

strcpy(stud->sname,sna);

strcpy(stud->name,na);

strcpy(stud->fname,fna);

strcpy(stud->phoneno,pno);

strcpy(stud->DOB,dob);

strcpy(stud->gender,gen);

stud->CGPI=cgpi;

strcpy(stud->branch,br);

strcpy(stud->year,yr);

stud->next=NULL;

if(start==NULL)

{

start=stud;

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*DATA INSERTED SUCCESSFULLY\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

}

else

{

temp=start;

while(temp->next!=NULL)

{

temp=temp->next;

}

temp->next=stud;

stud->prev=temp;

stud->next=NULL;

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*DATA INSERTED SUCCESSFULLY\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

}

}

void deleteID()

{

int val;

struct student \*del;

printf("\nEnter the id of the student to be removed : ");

scanf("%d",&val);

del=start;

if(isEmpty())

{

printf("\nNO RECORDS!");

}

else{

while(del->id!=val && del!=NULL)

{

del=del->next;

}

if(del==NULL)

{

printf("student with id %d does not exist...",val);

main();

}

else if(del==start && del->next==NULL)

{

printf("Data of student with id %d removed successfully ",del->id);

free(del);

}

else if(del==start)

{

start=del->next;

count--;

printf("Data of student with id %d removed successfully ",del->id);

free(del);

}

else

{

del->prev->next=del->next;

del->next->prev=del->prev;

printf("Data of student with id %d removed successfully ",del->id);

free(del);

}

}

}

void deleteName()

{

char val[20];

int valn;

int f=0;

struct student \*del;

printf("\nEnter the name of the student to be removed : ");

scanf("%s",&val);

del=start;

if(isEmpty())

{

printf("\nNO RECORDS!");

}

else{

while(del!=NULL)

{

if((strcmp(del->name,val)==0))

{

printf("\n/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*\n");

printf("\nROLL NO.: %d\nSURNAME.: %s\nNAME.: %s\nFATHER'S NAME.: %s\nPHONE NO.: %s\nD.O.B.: %s\nGENDER.: %s\nCGPI.: %f\nBRANCH.: %s\nYEAR.: %s\n",del->id,del->sname,del->name,del->fname,del->phoneno,del->DOB,del->gender,del->CGPI,del->branch,del->year);

f=1;

}

del=del->next;

}

}

if(del==NULL && f!=1)

{

printf("\nNO record with such name found!\n");

main();

}

else

deleteID();

}

void viewP()

{

int val;

struct student \*find;

printf("\nEnter the id of the student whose information is to be seen:");

scanf("%d",&val);

find=start;

while(find->id!=val && find!=NULL)

{

find=find->next;

}

if(find==NULL)

{

printf("student with id %d does not exist...",val);

main();

}

else

{

printf("Details of student with id %d",find->id);

printf("\n/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*\n");

printf("\nROLL NO.: %d\nSURNAME.: %s\nNAME.: %s\nFATHER'S NAME.: %s\nPHONE NO.: %s\nD.O.B.: %s\nGENDER.: %s\nCGPI.: %f\nBRANCH.: %s\nYEAR.: %s\n",find->id,find->sname,find->name,find->fname,find->phoneno,find->DOB,find->gender,find->CGPI,find->branch,find->year);

}

}

void save()

{

struct student \*p,\*q;

char temp[20];

float tempn;

int rn=0;

if(version==0)

{

rn=0;

}

if(isEmpty())

{

printf("\nNO RECORDS!");

}

else

{

for(p=start;p!=NULL;p=p->next)

{

for(q=start;q->next!=NULL;q=q->next)

{

if(strcmp(q->sname,q->next->sname)>0)

{

strcpy(temp,q->sname);

strcpy(q->sname,q->next->sname);

strcpy(q->next->sname,temp);

strcpy(temp,q->fname);

strcpy(q->fname,q->next->fname);

strcpy(q->next->fname,temp);

strcpy(temp,q->name);

strcpy(q->name,q->next->name);

strcpy(q->next->name,temp);

strcpy(temp,q->phoneno);

strcpy(q->phoneno,q->next->phoneno);

strcpy(q->next->phoneno,temp);

strcpy(temp,q->DOB);

strcpy(q->DOB,q->next->DOB);

strcpy(q->next->DOB,temp);

strcpy(temp,q->gender);

strcpy(q->gender,q->next->gender);

strcpy(q->next->gender,temp);

tempn=q->CGPI;

q->CGPI=q->next->CGPI;

q->next->CGPI=tempn;

strcpy(temp,q->branch);

strcpy(q->branch,q->next->branch);

strcpy(q->next->branch,temp);

strcpy(temp,q->year);

strcpy(q->year,q->next->year);

strcpy(q->next->year,temp);

}

else if(strcmp(q->sname,q->next->sname)==0)

{

if(strcmp(q->fname,q->next->fname)>0)

{

strcpy(temp,q->sname);

strcpy(q->sname,q->next->sname);

strcpy(q->next->sname,temp);

strcpy(temp,q->fname);

strcpy(q->fname,q->next->fname);

strcpy(q->next->fname,temp);

strcpy(temp,q->name);

strcpy(q->name,q->next->name);

strcpy(q->next->name,temp);

strcpy(temp,q->phoneno);

strcpy(q->phoneno,q->next->phoneno);

strcpy(q->next->phoneno,temp);

strcpy(temp,q->DOB);

strcpy(q->DOB,q->next->DOB);

strcpy(q->next->DOB,temp);

strcpy(temp,q->gender);

strcpy(q->gender,q->next->gender);

strcpy(q->next->gender,temp);

tempn=q->CGPI;

q->CGPI=q->next->CGPI;

q->next->CGPI=tempn;

strcpy(temp,q->branch);

strcpy(q->branch,q->next->branch);

strcpy(q->next->branch,temp);

strcpy(temp,q->year);

strcpy(q->year,q->next->year);

strcpy(q->next->year,temp);

}

else if((q->fname,q->next->fname)==0)

{

if((q->name,q->next->name)>0)

{

strcpy(temp,q->sname);

strcpy(q->sname,q->next->sname);

strcpy(q->next->sname,temp);

strcpy(temp,q->fname);

strcpy(q->fname,q->next->fname);

strcpy(q->next->fname,temp);

strcpy(temp,q->name);

strcpy(q->name,q->next->name);

strcpy(q->next->name,temp);

strcpy(temp,q->phoneno);

strcpy(q->phoneno,q->next->phoneno);

strcpy(q->next->phoneno,temp);

strcpy(temp,q->DOB);

strcpy(q->DOB,q->next->DOB);

strcpy(q->next->DOB,temp);

strcpy(temp,q->gender);

strcpy(q->gender,q->next->gender);

strcpy(q->next->gender,temp);

tempn=q->CGPI;

q->CGPI=q->next->CGPI;

q->next->CGPI=tempn;

strcpy(temp,q->branch);

strcpy(q->branch,q->next->branch);

strcpy(q->next->branch,temp);

strcpy(temp,q->year);

strcpy(q->year,q->next->year);

strcpy(q->next->year,temp);

}

}

}

}

}

for(p=start;p!=NULL;p=p->next)

{

rn++;

p->id=rn;

}

}

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*DATA SAVED SUCCESSFULLY\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

}

void saveF()

{

struct student \*p,\*q;

char temp[20];

float tempn;

if(version==0)

{

rn=0;

fptr = fopen("C:\\Users\\mehek\\CPROJECTS\\SDM1S\\data.txt","w");

}

for(p=start;p!=NULL;p=p->next)

{

fprintf(fptr,"\n%d",p->id);

fprintf(fptr,"\t%s",p->sname);

fprintf(fptr,"\t%s",p->name);

fprintf(fptr,"\t%s",p->fname);

fprintf(fptr,"\t%s",p->phoneno);

fprintf(fptr,"\t%s",p->DOB);

fprintf(fptr,"\t%s",p->gender);

fprintf(fptr,"\t%f",p->CGPI);

fprintf(fptr,"\t%s",p->branch);

fprintf(fptr,"\t%s",p->year);

}

fclose(fptr);

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*DATA SAVED SUCCESSFULLY\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

}

void sortCGPI()

{

struct student \*p,\*q;

char temp[20];

float tempn;

int tempid;

if(isEmpty())

{

printf("NO RECORDS!");

}

else if(start->next==NULL)

{

printf("CGPI is already sorted!");

}

else

{

for(p=start;p!=NULL;p=p->next)

{

for(q=start;q->next!=NULL;q=q->next)

{

if(q->CGPI<q->next->CGPI)

{

tempid=q->id;

q->id=q->next->id;

q->next->id=tempid;

strcpy(temp,q->sname);

strcpy(q->sname,q->next->sname);

strcpy(q->next->sname,temp);

strcpy(temp,q->fname);

strcpy(q->fname,q->next->fname);

strcpy(q->next->fname,temp);

strcpy(temp,q->name);

strcpy(q->name,q->next->name);

strcpy(q->next->name,temp);

strcpy(temp,q->phoneno);

strcpy(q->phoneno,q->next->phoneno);

strcpy(q->next->phoneno,temp);

strcpy(temp,q->DOB);

strcpy(q->DOB,q->next->DOB);

strcpy(q->next->DOB,temp);

strcpy(temp,q->gender);

strcpy(q->gender,q->next->gender);

strcpy(q->next->gender,temp);

tempn=q->CGPI;

q->CGPI=q->next->CGPI;

q->next->CGPI=tempn;

strcpy(temp,q->branch);

strcpy(q->branch,q->next->branch);

strcpy(q->next->branch,temp);

strcpy(temp,q->year);

strcpy(q->year,q->next->year);

strcpy(q->next->year,temp);

}

}

}

printf("\nThe data is sorted with respect to CGPI!");

viewAll();

}

main();

}

void modify()

{

struct student \*mod;

int ch,id;

char sna[20];

char na[20];

char fna[20];

char pno[20];

char dob[20];

char gen[20];

float cgpi;

char br[20];

char yr[20];

printf("Enter the student id whose information is to be modified : ");

scanf("%d",&id);

mod=start;

while(mod->id!=id && mod!=NULL)

{

mod=mod->next;

}

if(mod->id==id)

{

do

{

printf("\nDetails which can be updated:\n1. SURNAME.: %s\n2. NAME.: %s\n3. FATHER'S NAME.: %s\n4. PHONE NO.: %s\n5. D.O.B.: %s\n6. GENDER.: %s\n7. CGPI.: %f\n8. BRANCH.: %s\n9. YEAR.: %s\n10. EXIT\nEnter Choice: ",mod->sname,mod->name,mod->fname,mod->phoneno,mod->DOB,mod->gender,mod->CGPI,mod->branch,mod->year);

scanf("%d",&ch);

switch(ch)

{

case 1:

printf("\n\nSURNAME: ");

scanf("%s",&sna);

strcpy(mod->sname,sna);

break;

case 2:

printf("\nName: ");

scanf("%s",&na);

strcpy(mod->name,na);

break;

case 3:

printf("\nFather's Name: ");

scanf("%s",&fna);

strcpy(mod->fname,na);

break;

case 4:

printf("\nPhone no.: ");

scanf("%s",&pno);

strcpy(mod->phoneno,pno);

break;

case 5:

printf("\nD.O.B:");

scanf("%s",&dob);

strcpy(mod->DOB,dob);

break;

case 6:

printf("\nGender:");

scanf("%s",&gen);

strcpy(mod->gender,gen);

break;

case 7:

printf("\nCGPI: ");

scanf("%f",&cgpi);

mod->CGPI=cgpi;

break;

case 8:

printf("\nBranch: ");

scanf("%s",&br);

strcpy(mod->branch,br);

break;

case 9:

printf("\nYear: ");

scanf("%s",&yr);

strcpy(mod->year,yr);

break;

case 10:

break;

default:

printf("No such case exits!");

}

}while(ch!=10);

printf("\nStudent data successfully modified!\n");

printf("\nDetails updated:\n1. SURNAME.: %s\n2. NAME.: %s\n3. FATHER'S NAME.: %s\n4. PHONE NO.: %s\n5. D.O.B.: %s\n6. GENDER.: %s\n7. CGPI.: %f\n8. BRANCH.: %s\n9. YEAR.: %s\n10. EXIT\n",mod->sname,mod->name,mod->fname,mod->phoneno,mod->DOB,mod->gender,mod->CGPI,mod->branch,mod->year);

}

else

{

printf("student with id %d does not exist!",mod->id);

}

}

void viewAll()

{

struct student \*trav;

trav=start;

if(isEmpty())

{

printf("\nNO RECORDS!");

}

else

{

do

{

printf("\n/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*/\*");

printf("\nROLL NO.: %d\nSURNAME.: %s\nNAME.: %s\nFATHER'S NAME.: %s\nPHONE NO.: %s\nD.O.B.: %s\nGENDER.: %s\nCGPI.: %f\nBRANCH.: %s\nYEAR.: %s\n",trav->id,trav->sname,trav->name,trav->fname,trav->phoneno,trav->DOB,trav->gender,trav->CGPI,trav->branch,trav->year);

trav=trav->next;

}while(trav!=NULL);

}

}

void saveALL()

{

int id,i=1;

char sna[20];

char na[20];

char fna[20];

char pno[20];

char dob[20];

char gen[20];

float cgpi;

char br[20];

char yr[20];

struct student \*ptr;

struct student \*temp;

if ((fptr = fopen("C:\\Users\\mehek\\CPROJECTS\\SDM1S\\data.txt","r")) == NULL){

printf("Error! opening file");

// Program exits if the file pointer returns NULL.

exit(1);

}

else

while(i<=count)

{

ptr=(struct student\*)malloc(sizeof(struct student));

if(i==1)

{

ptr->next=NULL;

ptr->prev=NULL;

start=ptr;

}

else

{

temp=start;

while(temp->next!=NULL)

{

temp=temp->next;

}

temp->next=ptr;

ptr->prev=temp;

ptr->next=NULL;

}

fscanf(fptr,"%d",&id);

ptr->id=id;

fscanf(fptr,"%s",&sna);

strcpy(ptr->sname,sna);

fscanf(fptr,"%s",&na);

strcpy(ptr->name,na);

fscanf(fptr,"%s",&fna);

strcpy(ptr->fname,fna);

fscanf(fptr,"%s",&pno);

strcpy(ptr->phoneno,pno);

fscanf(fptr,"%s",&dob);

strcpy(ptr->DOB,dob);

fscanf(fptr,"%s",&gen);

strcpy(ptr->gender,gen);

fscanf(fptr,"%f",&cgpi);

ptr->CGPI=cgpi;

fscanf(fptr,"%s",&br);

strcpy(ptr->branch,br);

fscanf(fptr,"%s",&yr);

strcpy(ptr->year,yr);

i++;

}

version=0;

save();

}

int isEmpty()

{

if(start==NULL)

{

return 1;

}

else

return 0;

}

void main()

{

int ch;

int chn;

fptr = fopen("C:\\Users\\mehek\\CPROJECTS\\SDM1S\\data.txt","a");

if(fptr == NULL)

{

printf("Error!");

exit(1);

}

do

{

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* STUDENT DATA MANAGER \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\n1. Enter Data for New student.\n2. Save.\n3. View.\n4. Delete.\n5. Sort CGPI.\n6. Modify.\n7. Save ALL.\n8. Exit.\n\nEnter Choice : ");

scanf("%d",&ch);

switch(ch)

{

case 1:

enterD(); //function call for entering data

break;

case 2:

save();

break;

case 3:

if(isEmpty())

{

printf("\nNO RECORDS!");

}

else

{

do

{

printf("\n1. View ALL.\n2. View Particular Student.\n3. EXIT.\n Enter Choice:");

scanf("%d",&chn);

switch (chn)

{

case 1:

viewAll();

break;

case 2:

viewP();

break;

case 3:

break;

}

}while(chn!=3);

}

break; //displays all the data

case 4:

do

{

printf("\n1. Delete By ID.\n2. Delete By Name.\n3. Exit.\n Enter Choice:");

scanf("%d",&chn);

switch (chn)

{

case 1:

deleteID();

break;

case 2:

deleteName();

break;

case 3:

break;

}

}while(chn!=3);

break;

case 5:

sortCGPI();

break;

case 6:

modify();

break;

case 7:

saveALL();

break;

case 8:

saveF();

break;

}

}while(ch!=8);

}