STUDENT DATA ARRANGEMENT FRAME WORK

Team Members:

19BCE2179(Rithwik Reddy)
17BEE0274(Jaya Sabrish Reddy)
19BDS01160(Animesh Prakash)

Report submitted for the
Final Project Review of

Course Code: CSE2003 – DSA
Professor : Dr. Ilanthenral Kandasamy
Slot :B1
School :School of computer science and engineering

Introduction:

In software engineering, a Design and Analysis of Algorithm is a specific method of sorting out information in a PC with the goal that it very well may be utilized effectively. It can execute on or progressively specific Abstract Data Types (ADT), which are the methods for indicating the agreement of activities and their intricacy. In correlation, it is a solid usage of the agreement gave by an ADT

Information structures give a way to oversee a lot of information proficiently for utilizations, for example, huge databases and web ordering administrations. Normally, effective information structures are critical to planning proficient calculations. Some proper plan strategies and programming dialects stress information structures, as opposed to calculations, as the key sorting out factor in programming structure. Putting away and recovering can be completed on information put away in both principle memory and optional memory.

The usage of an information structures as a rule requires composing a lot of systems that make and control examples of that structure. The effectiveness of information structure can't be broke down independently from those tasks. This perception inspires the hypothetical idea of unique information type, an information structure that is characterized in a roundabout way by the activities that might be performed on it, and scientific properties of those tasks (counting their existence cost).

Literature Review Summary / Algorithm Review

APPLICATIONS OF LINKED LIST:

MUSIC PLAYER:

Music Player is best application of linked list where each track can be represented by a node of a linked list .Linked list is mostly used when we can traverse back one item or traverse one item to next. A Music player application maintains a Playlist where all the songs to be played are maintained in an order. If the user wants to add a song to the playlist, it can be added at the end of the list as the songs are always queued at last. So, here linked list insertion operation can be used. If the user wants to play the next track or skip the track then it can be done by traversing the linked list. In case if the user wants to play previous tracks also then doubly linked list can be used. In case if user wants to delete a song or track then deletion operation of a linked list can be used where a specific song or track(node of a linked list) is selected by a user to delete.

HYPERLINKS:

A hyperlink is a link in a website which directs to another website. Hyperlinks are needed frequently as they help users to make easy reference. While using hyperlinks there arises a need to move back to the previous link or move front to next link for best reference and understanding. So, in this application each website can be represented by a node of a linked list and also hyperlinks in a specified website can be represented by pointers to next website. By this user can easily traverse back and front of the websites he visited if doubly linked list is used.

GRAPHS:

Linked lists save most of the space than other data structures like arrays. Because, in arrays one should create memory statically. In linked lists one can create memory dynamically. In graphs only values of edges which are present need to be stored. If we use array to represent graph then the memory for the edges which are not present is also created unnecessarily. So through linked list this unnecessary memory can be saved. By using linked list a vertex in a graph can be represented by the first node of linked list and all the edges corresponding to the vertex are added to the linked list corresponding to particular vertex.

MEMORY MANAGEMENT IN OPERATING SYSTEM:

Memory blocks in the operating system are managed using linked lists. RAM contains a memory named as heap memory which contains free blocks and used blocks. Free blocks are those chunks of memory which are available for use. Used blocks are those which are currently in use. So there are two linked lists maintained, one for free blocks and other for used. Whenever a memory block is freed by some process then it is removed from linked list which is maintained for used blocks(from the end).So, linked list deletion operation can be applied here. At the same time when a memory block is freed it is added to the linked list which is maintained for the free blocks(to the end).Here, linked list insertion

operation can be applied. In similar way whenever a process requests a new memory block then a end node(block) corresponding to linked list which is maintained by free blocks is deleted and a new node(block) corresponding to linked list which is maintained by used blocks is added.

GALLERY APP

Photos apps used in laptops or smart phones contain facilities to go to next image or previous image. Here a photo can be represented as a node and next or previous photo can be represented as a pointer to next or previous photo. Whenever User wants to visit next image then linked list can be easily traversed.

So, doubly linked list is the best application for this. Also in Gallery App one can delete the specified photo which is similar to linked list deletion operation. Whenever a new picture is captured it is inserted at the end in an application which is much similar to linked list insertion operation.

UNDO and REDO operations

An undo and redo operation in many Microsoft applications like word and excel can be best corresponded to linked lists. In this each operation can be represented by a node. Whenever Undo is done a node can be traversed back and redo if done a node can be traversed next. So, here also doubly linked list can be used.

STACKS and QUEUES

The data structures stacks and queues can also be represented by linked lists. In this way space can be saved. By using linked list for stacks and queues memory can be managed efficiently by creating memory only for those data which are required. But when arrays are used if the required data is less than the array size, most of the memory is wasted.

POLYNOMIAL OPERATIONS

Linked lists can be used for polynomial operations where each term can be represented as a node. Each node contains coefficient and an exponent. So, operations like addition and subtraction of polynomials can be implemented by linked lists by traversing two lists simultaneously at the same time and applying operations on two lists.

RUNNING APPLICATIONS OF OS

All the running applications of operating system are kept in a circular linked list and the operating system gives a fixed time for each application which is running. So, Operating system keeps on iterating over this linked list until all applications are completed

DYNAMIC MEMORY ALLOCATION.

Dynamic memory allocation means allocating memory at run time. If memory is statically allocated it cant be changed further through out the program. But, if the memory is dynamically allocated it can be managed in the program itself by allocating and freeing it at required points of the program as dynamic memory allocation is done at the running time of the program. So by this memory can be effectively managed.

HASHING:

Hashing is one which helps to find out the data corresponding to specified index. If more than one record is used then linked list is maintained for such index.

MICROSOFT OFFICE POWER POINT

In Microsoft Office power point contains slides which can be corresponded to nodes in a linked list. One can insert a new slide any where and delete from anywhere.

Objective of the project:

We have an objective to create the code and algorithm about the Process used in the ranking system of the student such that if we entered the roll number his details to be presented by storing the student information.

Innovation component in the project

Here we are dealing with student result management system, where we use linked list as a major construction code involving sorting and searching. Where we create a record of a person which can be accessed by his roll number. Where here in this record we enter Name of the student, roll number and his Final CGPA and we display the output in the descending order of their CGPA. After this we can even add another record between the specified person or roll number. We can also delete the record and also search for a particular student record by the student's roll number.

Work done and implementation

Methodology adapted:

Mergesort takes the input list and treats it as a collection of small sorted lists. It makes logN passes along the list, and in each pass it combines each adjacent pair of small sorted lists into one larger sorted list. When a pass only needs to do this once, the whole output list must be sorted.

So here's the algorithm. In each pass, we are merging lists of size K into lists of size 2K. (Initially K equals 1)

This procedure only uses forward links, so it doesn't need a doubly linked list. If it does have to deal with a doubly linked list, the only place this matters is when adding another item to List.

Dataset used / Algorithms used :

a. Where from you are taking your dataset?

   From the github.

b. Is your project based on any other reference project (Stanford Univ. or MIT)?

   NO.

c. How does your project differ from the reference project?

   In the code used by us is familiar to us rather than the referece code.

Project Code:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct student{
char name[100];
char roll[10];
float grade;
struct student *next;
};
struct student *first=NULL,*last=NULL,*k;
struct student* SortedMerge(struct student* a, struct student* b);
```

```
void FrontBackSplit(struct student * source,
        struct student** frontRef, struct student** backRef);

void MergeSort(struct student** headRef)
{
  struct student* head = *headRef;
  struct student* a;
  struct student* b;


  if ((head == NULL) || (head->next == NULL))
  {
    return;
  }
  FrontBackSplit(head, &a, &b);

  MergeSort(&a);
  MergeSort(&b);

  *headRef = SortedMerge(a, b);
}
struct student* SortedMerge(struct student* a, struct student* b)
```

```
{

struct student* result = NULL;

  if (a == NULL)
     return(b);
  else if (b==NULL)
     return(a);


  if (a->grade >= b->grade)
  {
     result = a;
     result->next = SortedMerge(a->next, b);
  }
  else
  {
     result = b;
     result->next = SortedMerge(a, b->next);
  }
  return(result);
}
void FrontBackSplit(struct student* source,
```

```
        struct student** frontRef, struct student ** backRef)
{
 struct student * fast;
 struct student * slow;
 if (source==NULL || source->next==NULL)
 {
  *frontRef = source;
  *backRef = NULL;
 }
 else
 {
  slow = source;
  fast = source->next;

  while (fast != NULL)
  {
   fast = fast->next;
   if (fast != NULL)
   {
    slow = slow->next;
    fast = fast->next;
   }
```

```
    }
    *frontRef = source;
    *backRef = slow->next;
    slow->next = NULL;
  }
}

void create(int n)
{
int i;
first=(struct student*)malloc(sizeof(struct student));
printf("\nEnter the first name of the student:");
scanf("%s",first->name);
printf("\nEnter the roll number of the student:");
scanf("%s",first->roll);
printf("\nEnter the Final CGPA of the student:");
scanf("%f",&(first->grade));
first->next=NULL;
last=first;
for(i=1;i<n;i++) {
k=(struct student*)malloc(sizeof(struct student));
printf("\nEnter the first name of the student:");
```

```
scanf("%s",k->name);
printf("\nEnter the roll number of the student:");
scanf("%s",k->roll);
printf("\nEnter the Final CGPA of the student:");
scanf("%f", &(k->grade));
k->next=NULL;
last->next=k;
last=k;
}
}
void display()
{
struct student *t;
t=first;
while(t!=NULL)
{
printf("\nThe roll number of the student:%s",t->roll);
printf("\nFirst name of the student:%s",t->name);
printf("\nEnter the Final CGPA of the student:%f",t->grade);
t=t->next;
}
}
```

```
void insertafter()
{
char r[10];
int flag=0;
printf("\nEnter the roll number u wanna insert after that:");
scanf("%s",r);
struct student *t;
t=first;
while(t!=NULL)
{
if(strcmpi(r,t->roll)==0)
{
k=(struct student*)malloc(sizeof(struct student));
printf("\nEnter the first name of the student:");
scanf("%s",k->name);
printf("\nEnter the roll number of the student:");
scanf("%s",k->roll);
printf("\nEnter the Final CGPA of the student:");
scanf("%f", &(k->grade));
k->next=t->next;
t->next=k;
flag=1;
```

```
break;
}
t=t->next;
}
if(flag==0)
printf("\nThe element not found!!!");
}
void del()
{
struct student *back,*t,*k;
char r[10];
int flag=0;
printf("\nEnter the roll number u wanna delete:");
scanf("%s",r);
if(strcmpi(r,first->roll)==0)
{
first=first->next;
flag=1;
}
else
{
back=first;
```

```
k=first->next;
while(k!=NULL)
{
if(strcmpi(r,k->roll)==0)
{
back->next=k->next;
flag=1;
break;
}
}
back=k;
k=k->next;
}
if(flag==0)
printf("\nThe element not found!!!");
}
void search()
{
char r[10];
int flag=0;
printf("\nEnter the roll number u wanna search:");
scanf("%s",r);
```

```
struct student *t;
t=first;
while(t!=NULL)
{
if(strcmpi(r,t->roll)==0)
{
printf("\nThe roll number found in the list!!!\nHis name is %s",t);
flag=1;
break;
}t=t->next;
}
if(flag==0)
printf("\nThe roll number not in database!!");
}
int main()
{
int n,o;
while(o!=0)
{
printf("\nMENU\n");
printf("\nEnter 1 for creating database");
printf("\nEnter 2 for displaying database");
```

```
printf("\nEnter 3 for inserting an record after another");
printf("\nEnter 4 for deleting a record");
printf("\nEnter 5 for searching a record");
printf("\nEnter 0 for exit!");
printf("\nEnter the choice:");
scanf("%d",&o);
switch(o)
{
case 1:printf("\nEnter the maximum size of the database:");
scanf("%d",&n);
create(n);
break;
case 2:MergeSort(&first);display();break;
case 3:insertafter();break;
case 4:del();break;
case 5:search();break;
case 0:exit(0);break;
default:printf("\nYou have entered a wrong choice!!!");
}
}
}
```

Tools used:

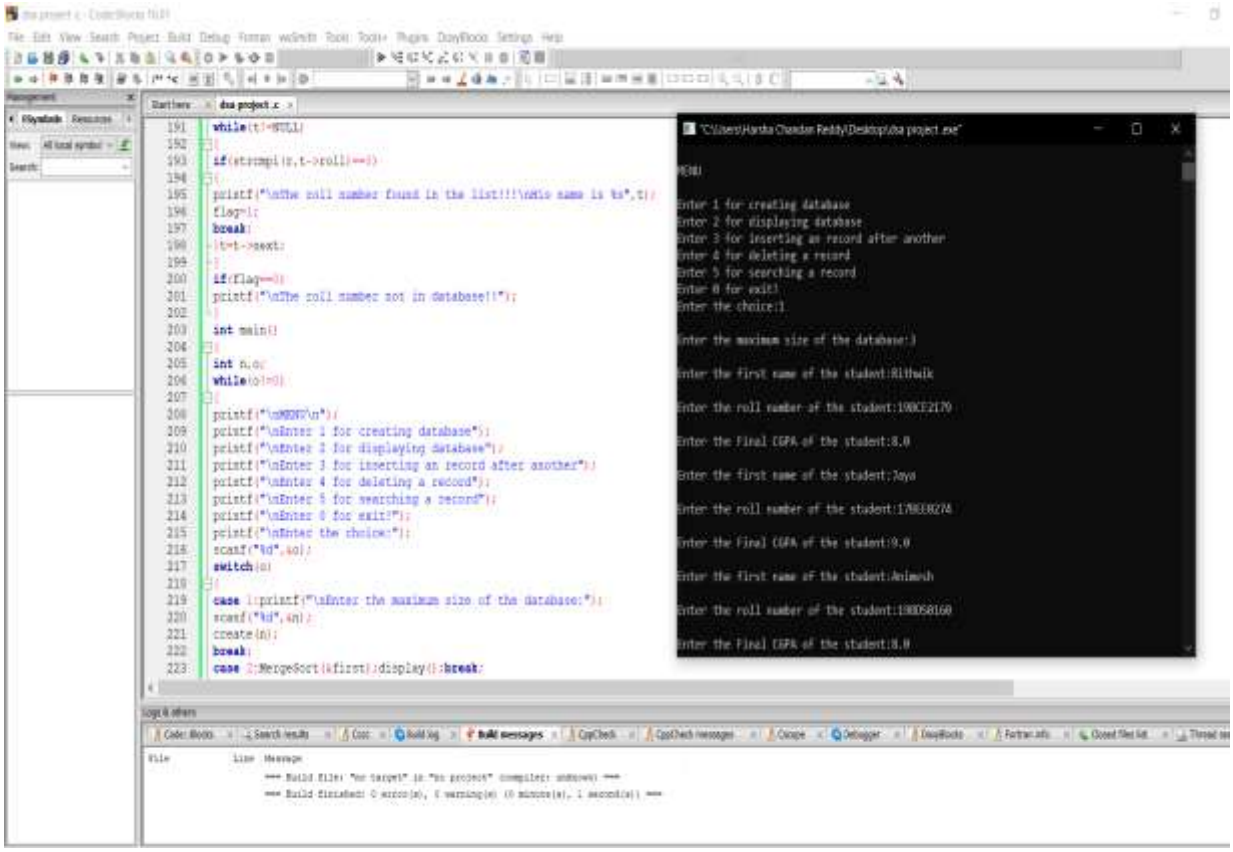
1.Codeblocks:

An integrated development environment (**IDE**) is a software application that provides comprehensive facilities to computer programmers for software development. An **IDE** normally consists of a source code editor, build automation tools and a debugger.

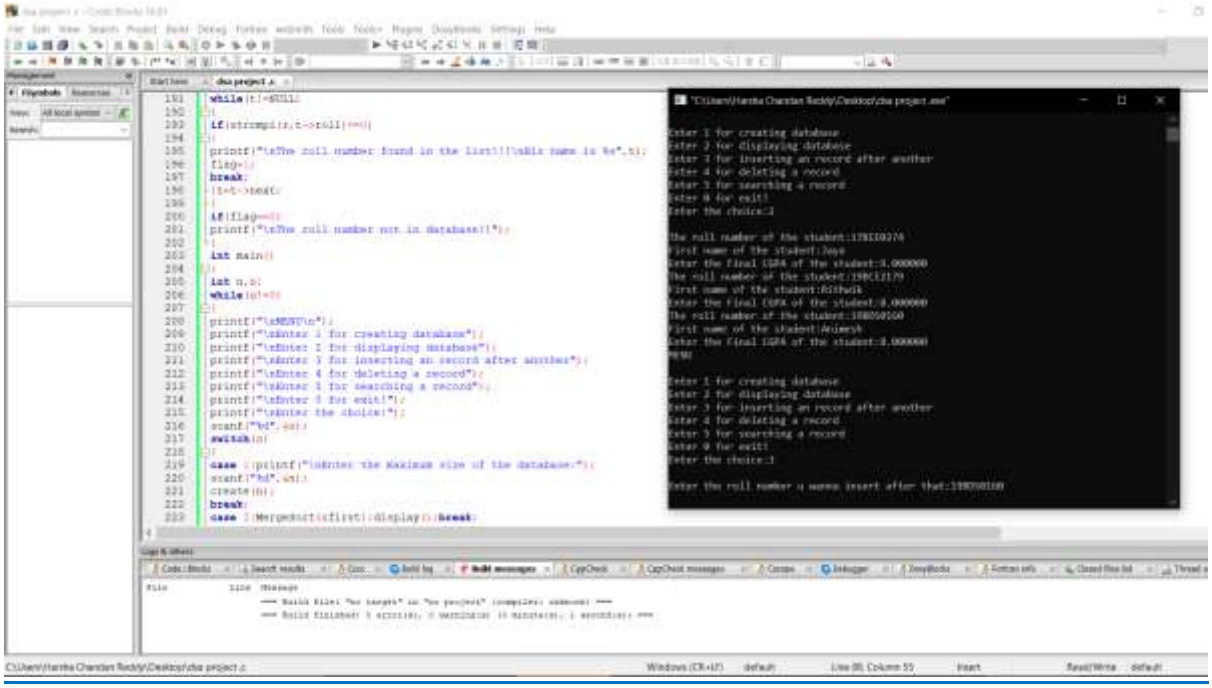
2.C language compiler:

Compiler is (generally) a program that takes your “source” code in one programming language (like C) and transforms is into “target” code *of the same semantics* in another programming language.

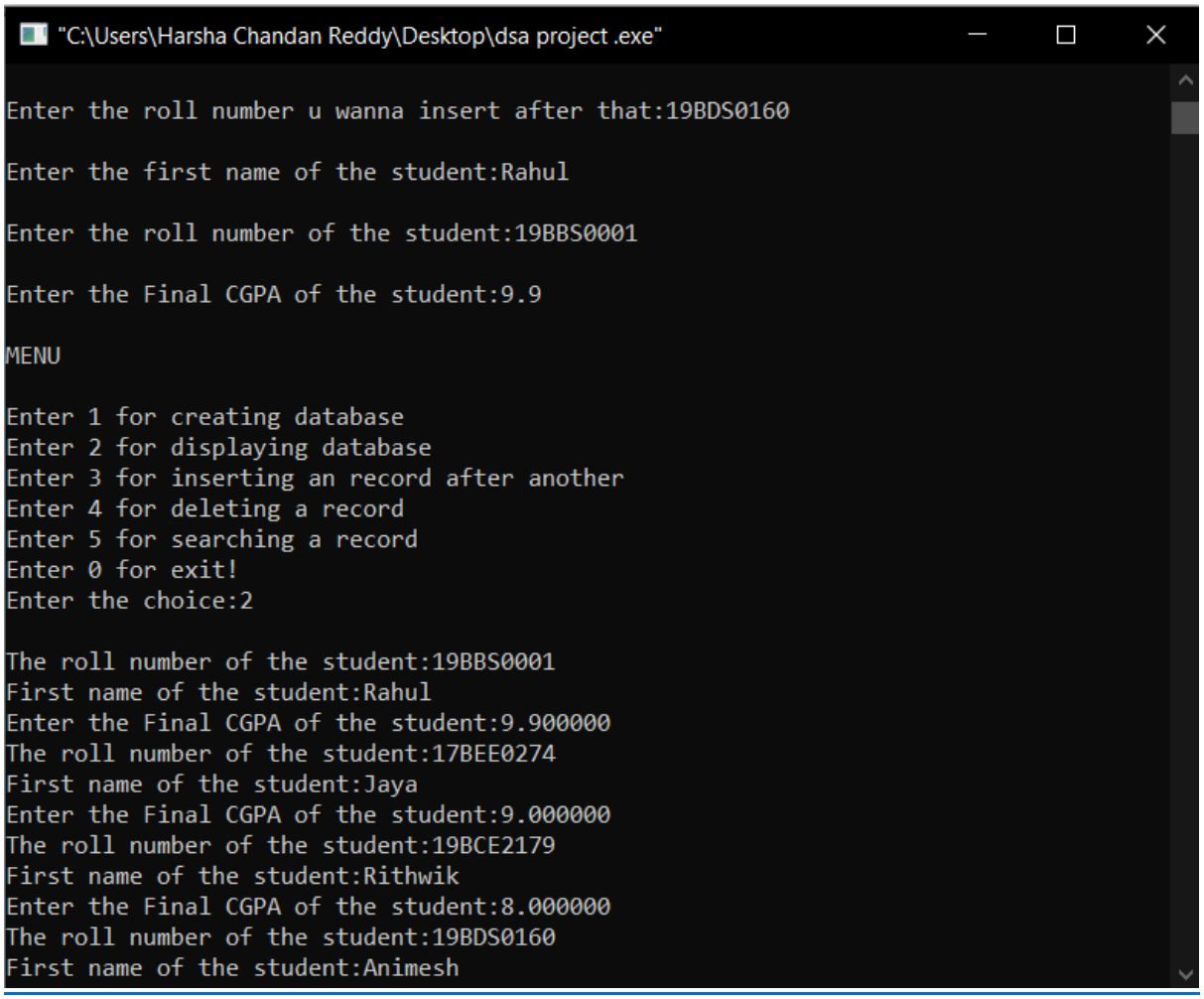
Results and discussion :



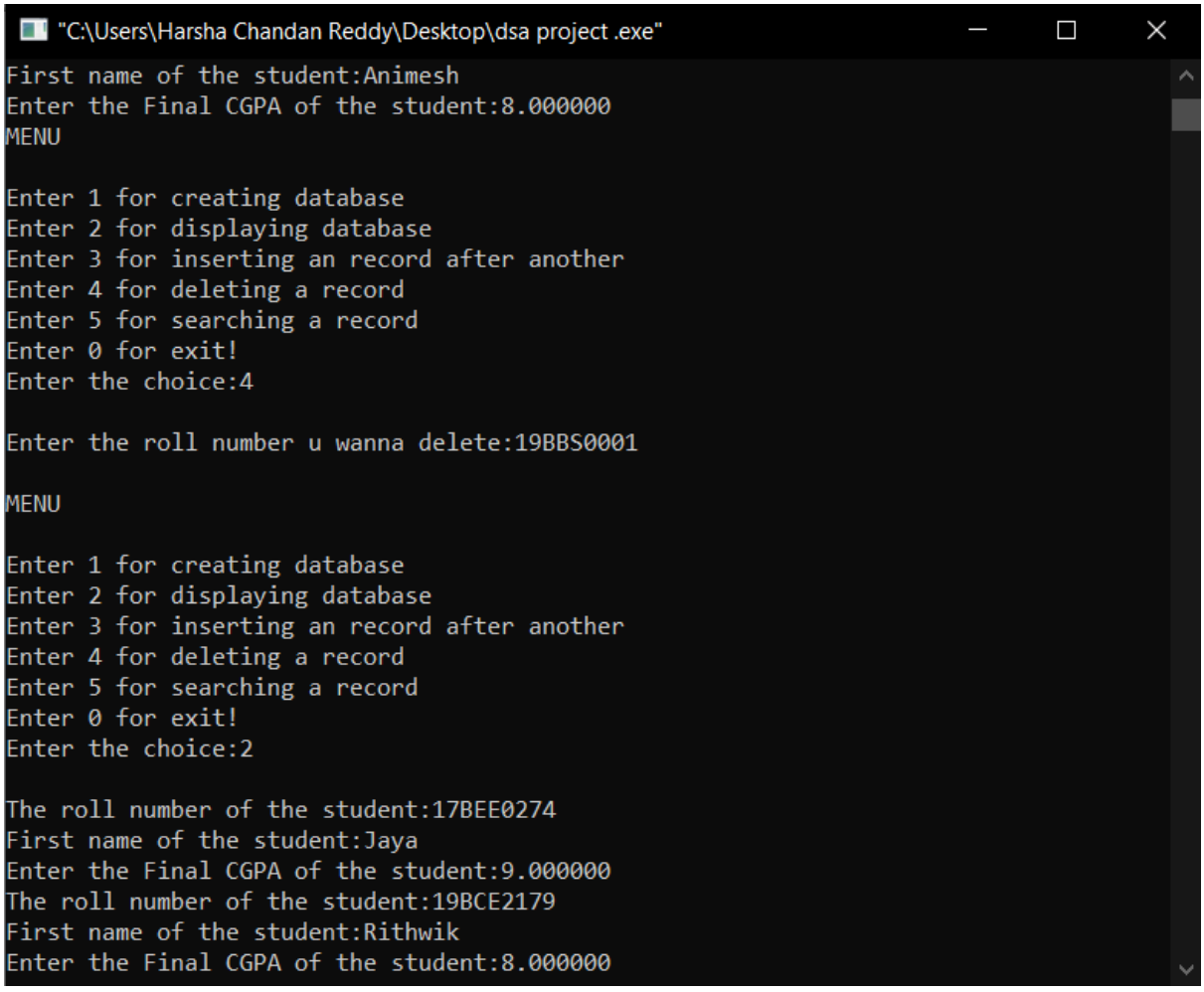
As we get the output we intrested in creating the data of the 3 people including the roll numbers,name,GPA.



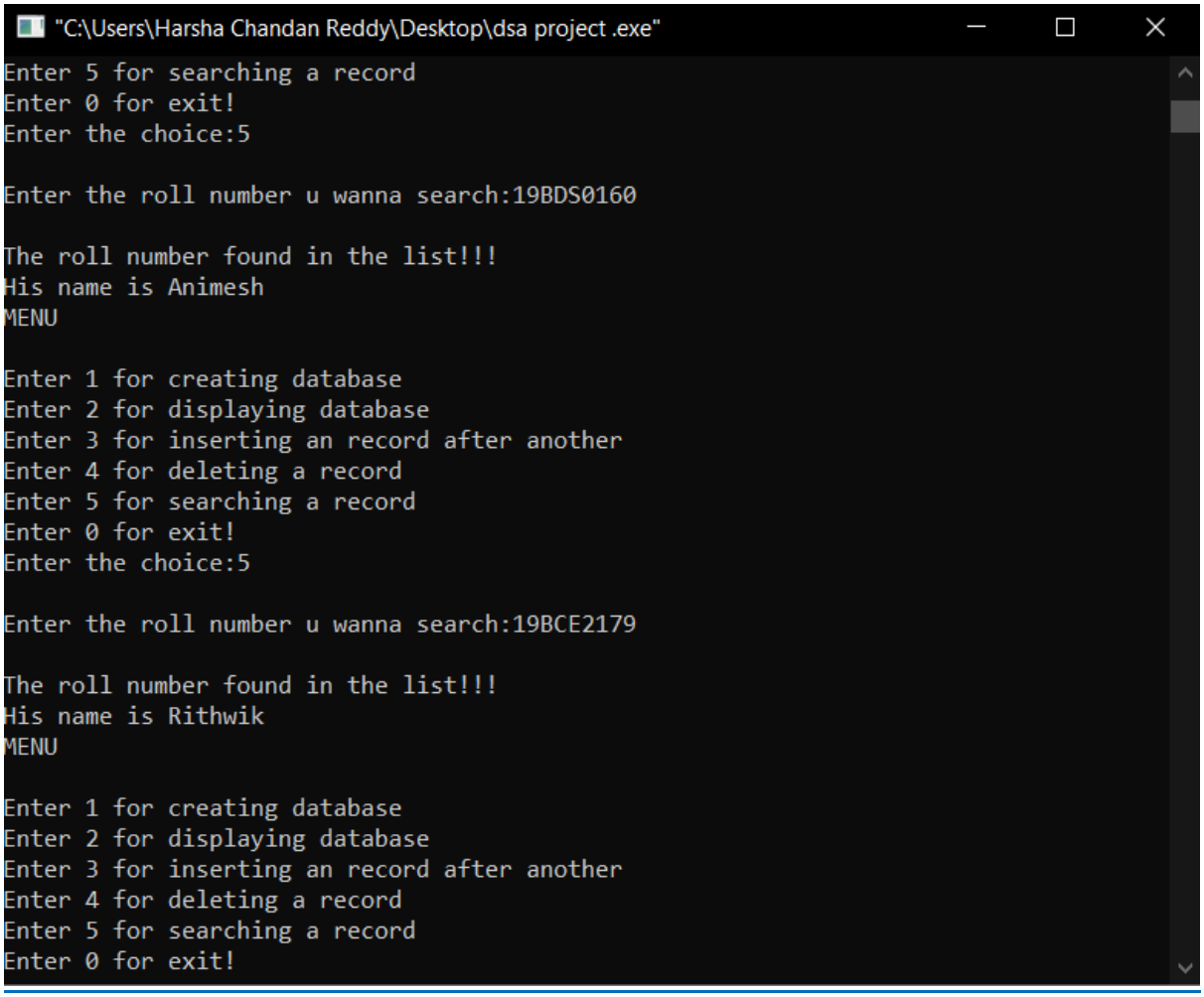
After entering the data of the 3 members we entered the option 2,that is to display the data of the members.



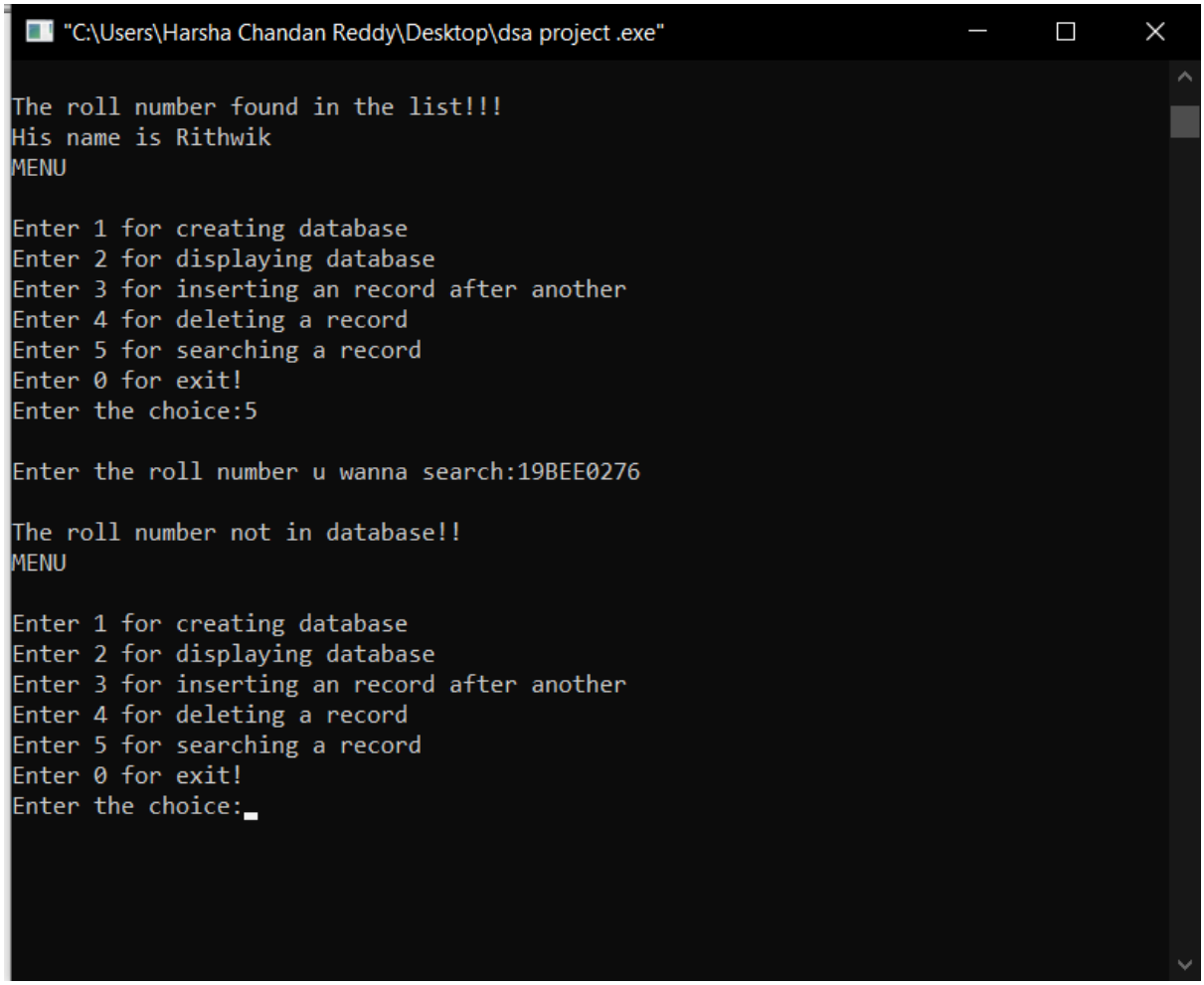
After displaying the data we would like to change the data of the of third person sowe selected option 3 and replaced by other data.



We thought to delete the third person record so we entered the third persons roll number and we deleted the record.



And at last we verified the data that had been presented at starting is there or not by entering search a record option.



And we verified all the persons data that we have entered in the record

PSEUDOCODE:

First we have created data about students result data base. In it we have to enter number of student data bases that to create.

After that we enter w.r.t the roll no. Of the particular student along with his name and GPA.

After that we can display the data in a decreasing order of GPA with the help of sorting function in which the data is divided into two halves and further divided into two until we have only two student data that can be compared and finally we merge the complete data in required order.

We can add data of any other student in between wherever we need, with the help of the function insert after, the working of the function is nothing but to insert data after any other student data as required with the help of roll no.

We can also delete the particular student data with the help of the roll no.

And finally we can search the required student data with the help of his roll no.

References:

1. Data Structure Through C In Depth- S.K. Srivastava

2. https:/www.w3schools.in/c-tutorial/file-handling

3. https:/www.tutorialpoint.com/cprogramming

4. https:/en.m.wikipedia.org/

5. www.stackoverflow.com

6. www.quora.com

7.www.kaggle.com

8.github