**HOSTEL COUNSELLING FOR FIRST YEAR STUDENTS**

**Project for**

**Data Structures And Algorithms – CSE 2003**

Submitted by:

16BCE0373 PULKIT GUPTA

Submitted to:

PROF. GOPINATH M.P

ASSOCIATE PROFESSOR

(SCOPE)

**SLOT: G1**



**VIT University, Vellore**

**Tamil Nadu - 632 014**

**March-2018**

# 

# 

# CERTIFICATE

This is to certify that the project work entitled **Hostel Counselling for first year students** that is being submitted by “**Pulkit Gupta**” for Data Structures And Algorithms (CSE2003) is a record of bonafide work done under my supervision.

Place:Vellore

Date :29/03/2018

Pulkit Gupta

**Signature of Faculty:**

**Prof. Gopinath M.P.**

# ACKNOWLEDGEMENTS

The completion of this undertaking could not have been possible without the participation and assistance of so many people whose names may not all be enumerated. Their contributions are sincerely appreciated and gratefully acknowledged. However, the group would like to express their deep appreciation and indebtedness to the following:

Prof. Gopinath M.P. for his endless support, kind and understanding spirit during our case presentation.

To all relatives, friends and others who in one way or another shared their support, either morally, financially or physically. Thank You.

Above all, to the Great Almighty, the author of knowledge and wisdom for his support.

**Thank You**

# ABSTRACT

Every year in the winter semester, hostel rooms are allotted to the students and the allotment is based on their ranks. The ranks are given by calculating their ncgpa. There are various stages involved like calculating the branch highest and the class highest ,which is followed by calculating their ncgpa and giving their ranks. Then, the hostel allotment is based on the queue concept (FIFO).

The branches are represented by a hash table each and the specific branch hash table consists of the classes of the given branches. Then the students are related to the given class by chaining through a given hash function. The registration number gives out the specific class of the student. The allotment is based on the queue concept. Here we make use of the various data structures like queue, hash tables and we also use collision resolution method of chaining.

# OVERVIEW

In this project, we choose and implement a NCGPA list for every student branch in a hash table and then generate a rank list for all the students in the university and then they will come in a queue according to their rank and select the room which they desire accordingly. **Objectives**

* To study the practical applications of hash table , queue , and linked list.
* To find a better and compact solution for a hostel room counselling session in VIT.

**Definitions:**

**Hash Table**: Hash Table is a data structure which store data in associative manner. In hash table, data is stored in array format where each data values has its own unique index value. Access of data becomes very fast if we know the index of desired data.

Thus, it becomes a data structure in which insertion and search operations are very fast irrespective of size of data. Hash Table uses array as a storage medium and uses hash technique to generate index where an element is to be inserted or to be located from.

Hashing is a technique to convert a range of key values into a range of indexes of an array. We're going to use modulo operator to get a range of key values. Consider an example of hashtable of size 20, and following items are to be stored. Item are in (key,value) format.

**Queue:** Queue is also an abstract data type or a linear data structure, in which the first element is inserted from one end called REAR(also called tail), and the deletion of exisiting element takes place from the other end called as FRONT(also called head). This makes queue as FIFO data structure, which means that element inserted first will also be removed first.

The process to add an element into queue is called Enqueue and the process of removal of an element from queue is called Dequeue.

# PROBLEM DESCRIPTION

1.The project is basically the allotment of hostel rooms based on the NCGPA of the students of different branches.

2.Separate hash tables have been used to represent different branches and each part of the hash table is a section.

3.According to the registration no. and branches in btech given by vit, students have been placed in the different sections of different hash tables

4.The students are related to a given section through chaining.

5.We have chosen the hash function such that we can relate a given student to his branch and class based on his registration number.

6.Once the student records have been entered, then the next step is to calculate the NCGPA of different students. For this first the CGPA of branch topper and class topper was found then the NCGPA was calculated.

7.Then ,we merged the different sections within a hash table and sorted them based on their NCGPA.

8.All the records in the different hash tables were merged and sorted to obtain the final list based on their NCGPA.

9.After this , the students will be arranged in a queue on the basis of their NCGPA rank.

10.After they are arranged , they are arranged in a queue which follows FIFO concept , the first element (student ) coming out of the queue will select the room and the element is deleted from the queue.

The reason we used hash table , we wanted to store the student details through a standard function , so hash table is the best for it. Collision has been resolved through chaining as we had to relate a large number of students with the specific part of the hash table and to save considerable space.We sorted and merged all the students just the way we do it in the merge sort as we already have the sorted arrays.

## BLOCK DIAGRAMS

**CSE**

**SECTION 1**

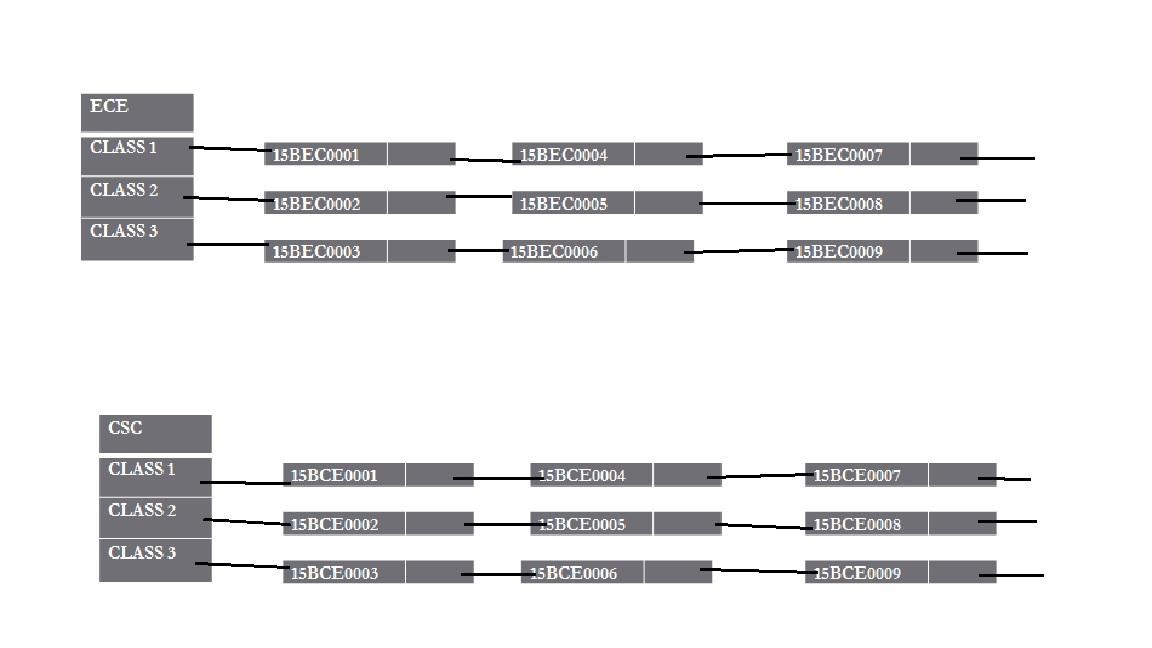
**SECTION 2**

**SECTION 3**

**SECTION 4**

**SECTION 5**

Each hash table corresponds to different branch and different elements of hash table correspond to different sections.



Students have been placed in different sections based on their registration numbers.

|  |
| --- |
| **15BEEE0232** |
| **15BCE0045** |
| **15BME0444** |
| **15BCE0012** |
| **15BPI0044** |
| **15BIT0987** |
| **15BEM0912** |
| **15BCH0121** |
| **15BCL764** |
| **15BBT0112** |
| **15BCB0099** |
| **15BCM4454** |
| **15BEC0035** |

Now the elements are set in a queue based on the NCGPA rank and the first element in will select the room first and and come out.

## PROJECT CODE

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include<conio.h>

int absentees[100]={}, presentees[100]={}, ablock\_1bed[20]={}, bblock\_2bed[10][2]={};

struct hash \*csetable=NULL; struct hash \*ecetable=NULL;

struct hash \*mechtable=NULL;

struct node

{

int rollno; float cgpa; float ncgpa; int rank;

char name[100]; struct node \*next;

};

struct node \*current; struct node \*temp;

struct node \*final=NULL;

struct hash

{

struct node \*head;

int count;

};

struct node \* createnode(int rollno, char \*name, float cgpa)

{

struct node \*newnode;

newnode = (struct node \*)malloc(sizeof(struct node));

newnode->rollno = rollno;

newnode->cgpa = cgpa;

newnode->ncgpa=0;

newnode->rank=0;

strcpy(newnode->name, name);

newnode->next = NULL;

return newnode;

}

void inserttohash(struct hash \*hashtable, int rollno, char \*name, float cgpa)

{

int hashindex = rollno % 3 +1;

struct node \*newnode = createnode(rollno, name, cgpa);

if (!hashtable[hashindex].head)

{

hashtable[hashindex].head = newnode;

hashtable[hashindex].count = 1;

current=newnode;

return;

}

else

{

struct node \*prev;

current=hashtable[hashindex].head;

if (newnode->cgpa > current->cgpa )

{

newnode->next=current;

hashtable[hashindex].head=newnode;

return;

}

else

{

while (current!=NULL)

{

if (newnode->cgpa <= current->cgpa)

{

prev=current;

current=current->next;

}

else

break;

}

newnode->next=prev->next;

prev->next=newnode;

hashtable[hashindex].count++;

return;

}

}

}

void deletefromhash(struct hash \*hashtable, int rollno)

{

int hashindex = rollno % 3 +1, flag = 0;

struct node \*temp, \*mynode;

mynode = hashtable[hashindex].head;

if (!mynode)

{

printf("Given data is not present in hash Table!!\n");

return;

}

temp = mynode;

while (mynode != NULL)

{

if (mynode->rollno == rollno)

{

flag = 1;

if (mynode == hashtable[hashindex].head)

hashtable[hashindex].head = mynode->next;

else

temp->next = mynode->next;

hashtable[hashindex].count--;

free(mynode);

break;

}

temp = mynode;

mynode = mynode->next;

}

if (flag)

printf("Data deleted successfully from Hash Table\n");

else

printf("Given data is not present in hash Table!!!!\n");

return;

}

void searchinhash(struct hash\* hashtable, int rollno)

{

int hashindex = rollno % 3 +1, flag = 0;

struct node \*mynode; mynode = hashtable[hashindex].head;

if (!mynode)

{

printf("Search element unavailable in hash table\n");

return;

}

while (mynode != NULL)

{

if (mynode->rollno == rollno)

{

printf("Roll no : %d\n", mynode->rollno);

printf("Name : %s\n", mynode->name);

printf("Cgpa : %f\n", mynode->cgpa);

flag = 1;

break;

}

mynode = mynode->next;

} if (!flag)

printf("Search element unavailable in hash table\n");

return;

}

void display(struct hash\* hashtable)

{

struct node \*mynode;

int i;

for (i = 1; i <= 3; i++)

{

if (hashtable[i].count == 0)

continue;

mynode = hashtable[i].head;

if (!mynode) continue;

printf("\nStudents in section %d of branch:\n", i);

printf("Rollno Name cgpa ncgpa \n");

printf("---------------------------------------------\n");

while (mynode != NULL)

{

printf("%-10d", mynode->rollno);

printf("%-13s", mynode->name);

printf("%f\t", mynode->cgpa);

printf("\t%f\n", mynode->ncgpa);

mynode = mynode->next;

}

}

return;

}

void calculatencgpa (struct hash \*hashtable ,struct node \*branchtopper)

{

struct node \*ctopper;

struct node \*btopper;

struct node \*curr;

int i;

float ncgpa;

btopper=branchtopper;

for (i=1; i<=3; i++)

{

ctopper=hashtable[i].head;

curr=hashtable[i].head;

while (curr!=NULL)

{

curr->ncgpa=(curr->cgpa \* ctopper->cgpa)/btopper->cgpa;

curr=curr->next;

}

}

return;

}

void findtopper(struct hash \*hashtable, int p)

{

int i, flag=0;

//int hashIndex = rollno % 3 +1;

struct node \*mynode;

struct node\* branchtopper;

for (i=1; i<=3; i++)

{

if (hashtable[i].head==NULL)

flag=1;

}

if (flag==1)

return;

branchtopper=hashtable[1].head;

for (i=2; i<=3; i++)

{

mynode = hashtable[i].head;

if (branchtopper->cgpa < mynode->cgpa )

{

branchtopper=mynode;

}

}

printf("\n\nDetails of branch %d topper is: \n", p);

printf("\nrollno: %d", branchtopper->rollno );

printf("\ncgpa: %f\n", branchtopper->cgpa );

calculatencgpa (hashtable, branchtopper);

return;

}

void displaylist(struct node\* abc)

{

struct node\* ptr=NULL; ptr=abc;

printf("Rollno Name cgpa ncgpa \n");

printf("---------------------------------------------\n");

while (ptr!=NULL)

{

printf("%-10d", ptr->rollno);

printf("%-13s", ptr->name);

printf("%f\t", ptr->cgpa);

printf("\t%f\n", ptr->ncgpa);

ptr=ptr->next;

}

printf("\n\n list has been printed\n");

return;

}

struct node\* merge(struct node\* temp1, struct node\* temp2)

{

struct node\* combinelist= NULL;

combinelist=(struct node\*)malloc(sizeof(struct node\*));

if (temp1 == NULL)

return(temp2);

else if (temp2 == NULL)

return(temp1);

if ( temp1->ncgpa >= temp2 ->ncgpa)

{

combinelist=temp1;

combinelist->next=merge(temp1->next, temp2);

} else

{

combinelist=temp2;

combinelist->next=merge(temp1,temp2->next);

}

return(combinelist);

}

struct node \*copyll(struct node \*start1)

{

struct node \*start2=NULL,\*previous=NULL;

while(start1!=NULL)

{

struct node \* temp = (struct node \*) malloc (sizeof(struct node));

temp->rollno = start1->rollno;

temp->cgpa = start1->cgpa;

temp->ncgpa=start1->ncgpa;

strcpy(temp->name, start1->name);

temp->next= NULL;

if(start2==NULL)

{

start2=temp;

previous=temp;

}

else

{

previous->next=temp;

previous=temp;

}

start1=start1->next;

}

return start2;

}

struct node\* ncgpalist(struct hash \*hashtable)

{

struct node \*list1=NULL, \*list2=NULL, \*list3=NULL, \*list4=NULL, \*branchlist=NULL;

struct node \*ptr1=NULL, \*ptr2=NULL, \*ptr3=NULL, \*ptr4=NULL;

//printf("\nflag 1");

list1=hashtable[1].head;

list2=hashtable[2].head;

list3=hashtable[3].head;

//printf("\nflag2");

ptr1=copyll(list1);

ptr2=copyll(list2);

ptr3=copyll(list3);

// printf("\nflag3");

list4=merge(ptr1, ptr2);

ptr4=copyll(list4);

branchlist=merge(ptr3, ptr4);

return branchlist;

}

struct node\* mergeall (struct node\* branch1, struct node\* branch2, struct node\* branch3)

{

struct node \*templist=NULL, \*finallist=NULL;

templist=merge(branch1, branch2);

finallist=merge(templist, branch3);

printf("\n\n Displaying final ncgpa list.....\n\n\n");

displaylist(finallist);

return finallist;

}

int checkrollno (struct hash \*hashtable, int rollno, int check)

{

int hashvalue=rollno%3 + 1;

struct node \*temp;

temp=hashtable[hashvalue].head;

if (!temp)

{

return check;

}

while(temp!=NULL)

{

if (temp->rollno==rollno)

{

printf("\n student record exists .....enter data again");

check=0;

return check;

}

temp=temp->next;

}

return check;

}

int selectbranch(int rollno, int n, int op)

{

int i,check, flag;

if (rollno>0 && rollno<=n)

{

check=1;

if (op==1)

flag=checkrollno(csetable,rollno, check);

printf("\n cse branch student");

}

else if (rollno>n && rollno<=2\*n)

{

check=2;

if (op==1)

flag=checkrollno(ecetable, rollno, check);

printf("\n ece branch student");

}

else if (rollno>2\*n && rollno<=3\*n)

{

check=3;

if (op==1)

flag=checkrollno(mechtable, rollno, check);

printf("\n mech branch student");

}

else

{

printf("\n Invalid rollno has been entered....enter again\n");

}

if (op==1)

return flag;

else

return check;

}

int printlist()

{

struct node \*ptr; int x, rank=1, flag=0; ptr=final;

if (ptr==NULL)

{

printf("\n\n SERVER ERROR !!");

flag=1;

}

if (flag==1)

return;

printf("\n\nRank Rollno NCGPA\n");

printf("---------------------------------\n");

while (ptr!=NULL)

{

ptr->rank=rank;

printf("\n%d",ptr->rank);

printf("\t%d",ptr->rollno );

printf("\t%f", ptr->ncgpa);

ptr=ptr->next;

rank=rank+1;

}

printf("\n\n full list has been printed\n");

return rank;

}

void tokenlist ()

{

int i=1;

struct node \*current= NULL;

printf("\n\n The student rankers taking part in counselling are: ");

current=final;

while (current!=NULL)

{

presentees[i]=current->rank;

current=current->next;

i++;

}

i=1;

printf("\n");

while (presentees[i]!=0)

{

printf("%d ",presentees[i]);

i++;

}

}

int findstudent( int arank)

{

struct node \*prev=NULL;

struct node \*current=NULL;

struct node \*temp=NULL;

current=final;

while (current!=NULL)

{

if (current->rank==arank)

{

prev->next=current->next;

temp=current;

current=current->next;

free(temp);

printf("Element has been found ");

return 1;

}

else

{

prev=current;

current=current->next;

}

}

printf("\n Roll number not found....");

return 0;

}

void registration( int totalstud)

{

struct node\* prev=NULL;

struct node\* current=NULL;

current=final;

int i=1, absentrank=0, found=0;

printf("\nTotalstud is: %d", totalstud-1);

printf("\n Enter the rankers not attending counselling: ");

while (i<=totalstud && absentrank!=-1)

{

scanf("%d", &absentrank);

if (absentrank==-1)

{

i=totalstud+1;

}

else

{

found=findstudent(absentrank);

if (found==1)

{

absentees[i]=absentrank;

i++; }

else

printf("\n go on enter again\n");

}

}

i=1;

printf("\nso now the ranks not attending counselling are: \n");

while (i<=100)

{

if (absentees[i]!=0)

{

printf("%d ", absentees[i]);

i++;

}

else if (absentees[i]==0)

goto v;

}

v: tokenlist() ;

}

int searchroommate(int rank)

{

int i=1, found=0;

while (absentees[i]!=0)

{

if (absentees[i]==rank)

{

found=1;

absentees[i]=-1;

break;

}

if ((absentees[i]==0 || absentees[i]==-1) && !(0<absentees[i]<30))

found=-1;

i++;

}

return found;

}

int chooseroom(int choice, int front)

{

int i=0, allot=0, found=0, roommaterank;

if (choice==1)

{

for (i=0; i<20; i++)

{

if (ablock\_1bed[i]==0)

{

ablock\_1bed[i]=front;

printf("CONGRATULATIONS !! Student Rank %d has been alloted 1 bed AC room in A block ",presentees[front]);

allot=1;

break;

}

}

if (allot==0)

printf("\nSorry, A Block is full");

}

if (choice==2)

{

z:

printf("Enter ncgpa rank of roommate: ");

scanf("%d", &roommaterank );

found=searchroommate(roommaterank);

if (found==0)

{

printf("\nRoommate not found!!");

goto z;

}

if (found==-1)

{

printf("No roommate available ");

allot=0;

return allot;

}

for (i=0; i<10; i++)

{

if (bblock\_2bed[i][0]==0)

{

bblock\_2bed[i][0]=front;

bblock\_2bed[i][1]=roommaterank;

printf("CONGRATULATIONS !! Student Rank %d and %d have been alloted 2 bed AC rom in B Block", presentees[front], roommaterank);

allot=1;

break;

}

}

if (allot==0)

printf("Sorry, B Block is full");

}

return allot;

}

int main()

{

int n, ch, rollno; float cgpa; char name[100];

printf("Number of students in each branch is equal \nEnter the number of students in a branch: ");

scanf("%d", &n);

int p, branch, branchno, operation, mycounter=0, ans, totalstud, choice, sec=4;

csetable = (struct hash \*)calloc(sec, sizeof (struct hash));

ecetable = (struct hash \*)calloc(sec,sizeof (struct hash));

mechtable = (struct hash \*)calloc(sec, sizeof (struct hash));

struct node \*b1list=NULL, \*b2list=NULL, \*b3list=NULL;

while (mycounter<=(3\*n))

{

printf("\n\n1. Insertion\t\t2. Deletion\n");

printf("3. Searching\t\t4. Display\n5. Calculate NCGPA\t6. Exit\n");

printf("\n\nEnter your choice: ");

scanf("%d", &ch);

switch (ch)

{

case 1: printf("Enter the rollno value: ");

scanf("%d", &rollno);

getchar();

printf("Name: ");

fgets(name, 100, stdin);

name[strlen(name) - 1] = '\0';

a: printf("cgpa: ");

scanf("%f", &cgpa);

if (cgpa<0 || cgpa>10)

goto a;

operation=1;

branch=selectbranch(rollno, n, operation);

if (branch==1)

{

inserttohash(csetable, rollno, name, cgpa);

mycounter++;

}

else if (branch==2)

{

inserttohash(ecetable, rollno, name, cgpa);

mycounter++;

}

else if (branch==3)

{

inserttohash(mechtable, rollno, name, cgpa);

mycounter++;

}

break;

case 2: system("cls");

printf("\nEnter the rollno to perform deletion: ");

scanf("%d", &rollno);

operation=2;

branch=selectbranch(rollno, n, operation);

if (branch==1)

{

deletefromhash(csetable, rollno);

mycounter--;

}

else if (branch==2)

{

deletefromhash(ecetable, rollno);

mycounter--;

}

else if (branch==3)

{

deletefromhash(mechtable, rollno);

mycounter--;

}

break;

case 3: system("cls");

printf("\nEnter the rollno to search:");

scanf("%d", &rollno);

branch=rollno/n+1;

if (branch==1)

searchinhash(csetable, rollno);

else if (branch==2)

searchinhash(ecetable, rollno);

else if (branch==3)

searchinhash(mechtable, rollno);

break;

case 4: system("cls");

printf("Displaying all records ");

printf("\n--------------------------------------------------------");

printf("\n\nDisplaying cse students \n");

display(csetable);

printf("\n--------------------------------------------------------");

printf("\n\nDisplaying ece students \n");

display(ecetable);

printf("\n--------------------------------------------------------");

printf("\n\nDisplaying mech students \n");

display(mechtable);

printf("\n--------------------------------------------------------");

printf("\n\n\nPress any key to continue ");

getch();

system("cls");

break;

case 5: printf("Want to proceed ahead, press 1 to proceed..");

scanf("%d", &ans);

if (ans==1)

mycounter=3\*n+1;

break;

case 6: printf("\nAre you sure you want to exit .... then press 1 to exit");

scanf("%d", &ans);

if (ans==1)

exit(0);

else

break;

default: printf("U have entered wrong option!!\n");

break;

}

}

system("pause");

system("cls");

printf("\nDisplaying branch topper details...\n");

branchno=1;

findtopper(csetable, branchno);

branchno=2;

findtopper(ecetable, branchno);

branchno=3;

findtopper(mechtable, branchno);

printf("\n\n");

system("pause");

system("cls");

printf("\nCreating the NCGPA lists.......Press any key to display the sorted ncgpa lists \n\n");

getch();

b1list=ncgpalist(csetable);

b2list=ncgpalist(ecetable);

b3list=ncgpalist(mechtable);

printf("\n\nDisplaying sorted ncgpa list of cse students\n\n");

displaylist(b1list);

printf("\n\nDisplaying sorted ncgpa lists of ece students\n\n");

displaylist(b2list);

printf("\n\n Displaying the sorted ncgpa lists of mech students\n\n");

displaylist(b3list);

printf("\n\n\n Next step is to merge all these 3 lists.....\n");

getch();

system("cls");

printf("\n\n Displaying merged list...\n\n");

final=mergeall(b1list, b2list, b3list);

system("pause");

system("cls");

printf(" Displaying NCGPA rank list");

totalstud=printlist();

printf("\n Do you want to proceed to room counselling day, if yes press 1 ....or else exit from program: ");

scanf("%d", &ans);

if (ans!=1)

exit(0);

system("cls");

printf("\n\nWelcome to room allotment counselling\n");

printf("\nProceed for registration: \n");

registration( totalstud);

printf("\n\n");

printf("\nTime to select rooms");

printf("\n\nRoom Type \t Total Rooms \t Choice No.");

printf("\n1 AC A Block \t\t 10 \t\t 1");

printf("\n2 AC B Block \t\t 10 \t\t 2");

int front=1;

while (presentees[front]!=0)

{

int allot=0, choice;

while (allot==0)

{

printf("\n\nStudent rank %d Choose your room type : ", presentees[front]);

scanf("%d", &choice);

if (choice==1 || choice ==2)

{

allot=chooseroom(choice, front);

if (allot==1)

{

front+=1;

}

}

}

}

printf("\n\nCOUNSELLING PROCESS OVER !!");

printf("\n..........THE END.............");

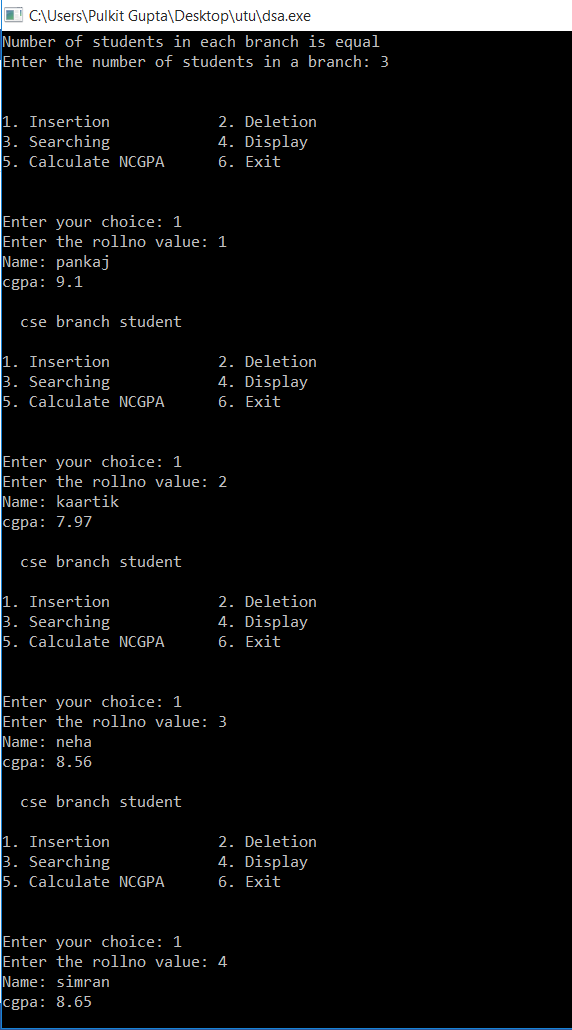
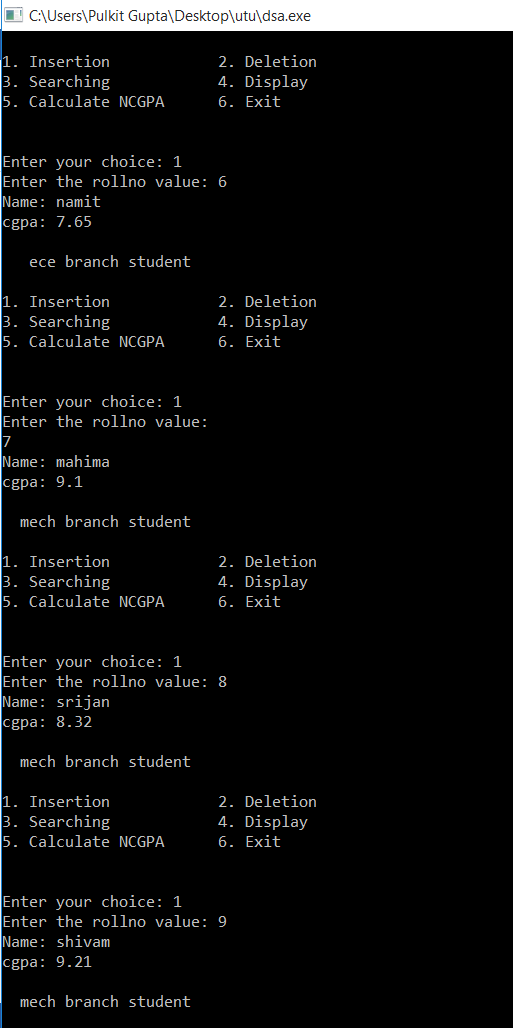
return 0;

getch();

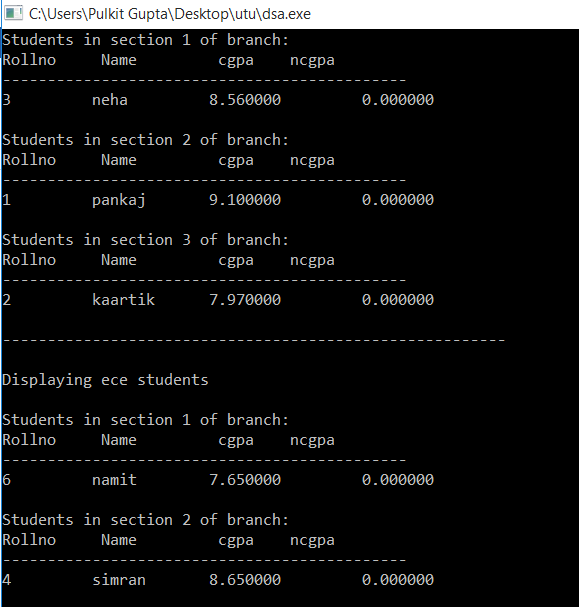
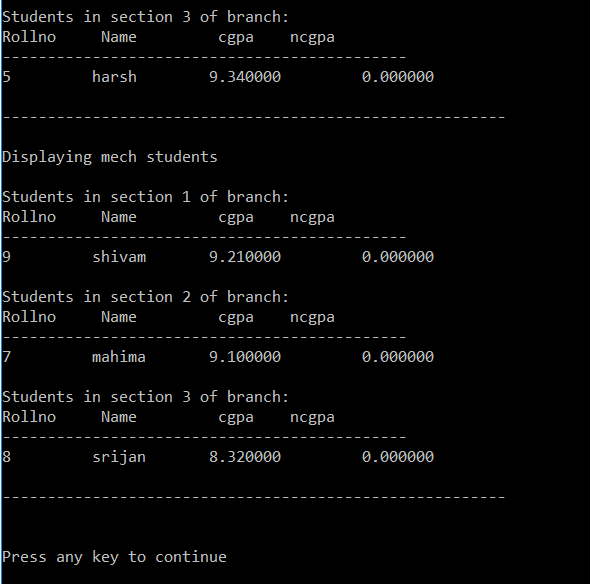
}

# SCREENSHOTS

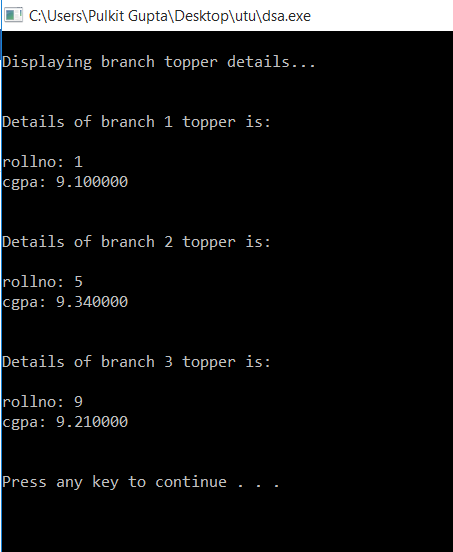
Taking student data as input



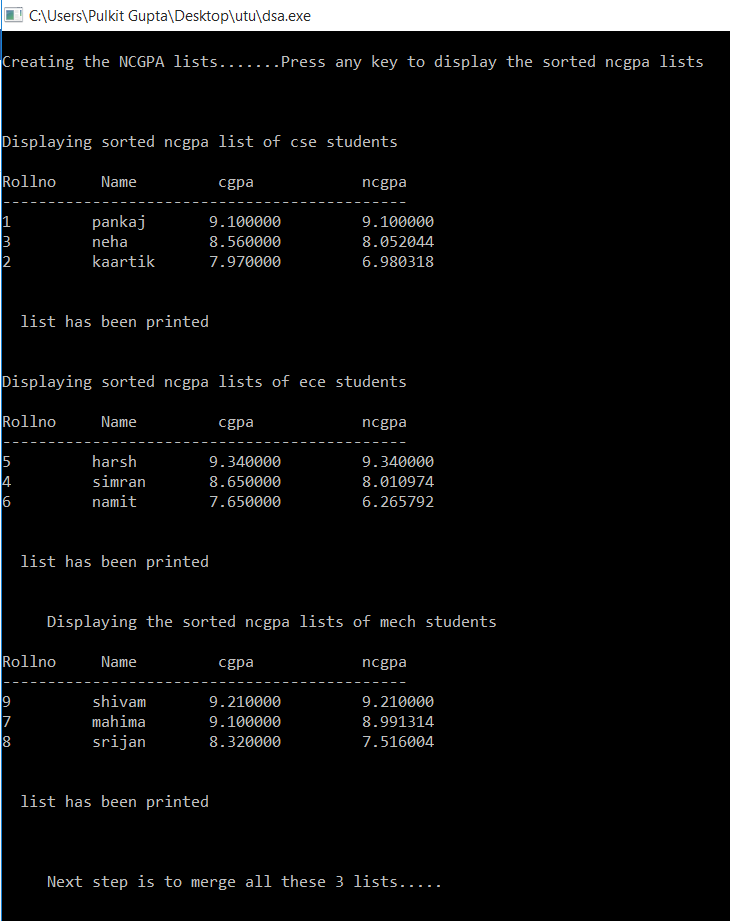
Displaying student data (branch and section wise)



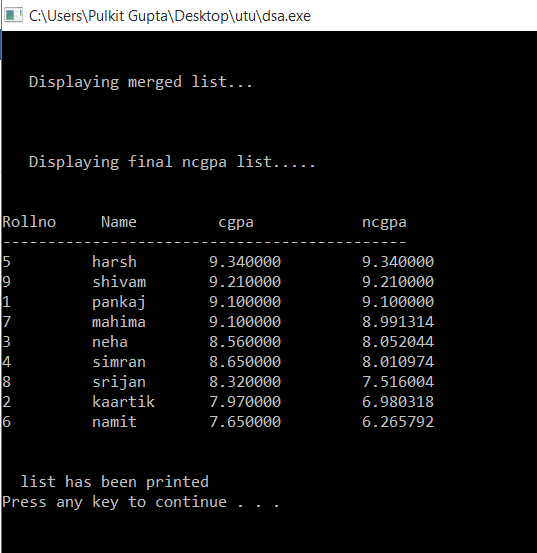
Displaying branch topper

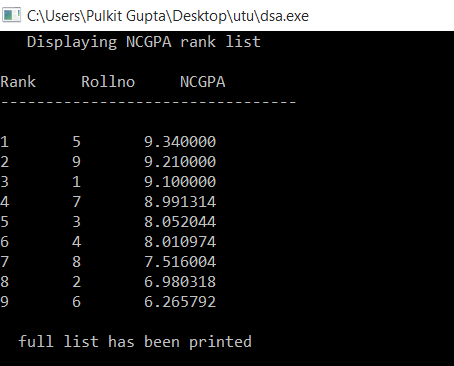


After calculating ncgpa, arranging branch wise

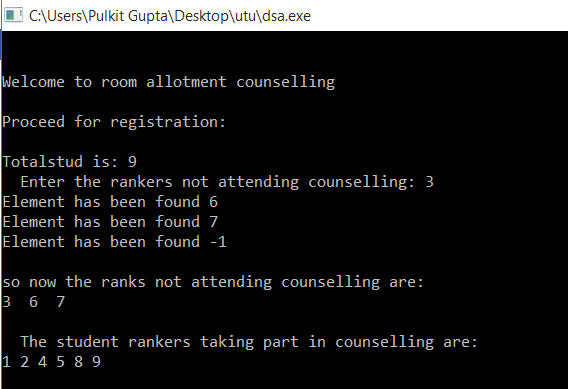


Merging all list



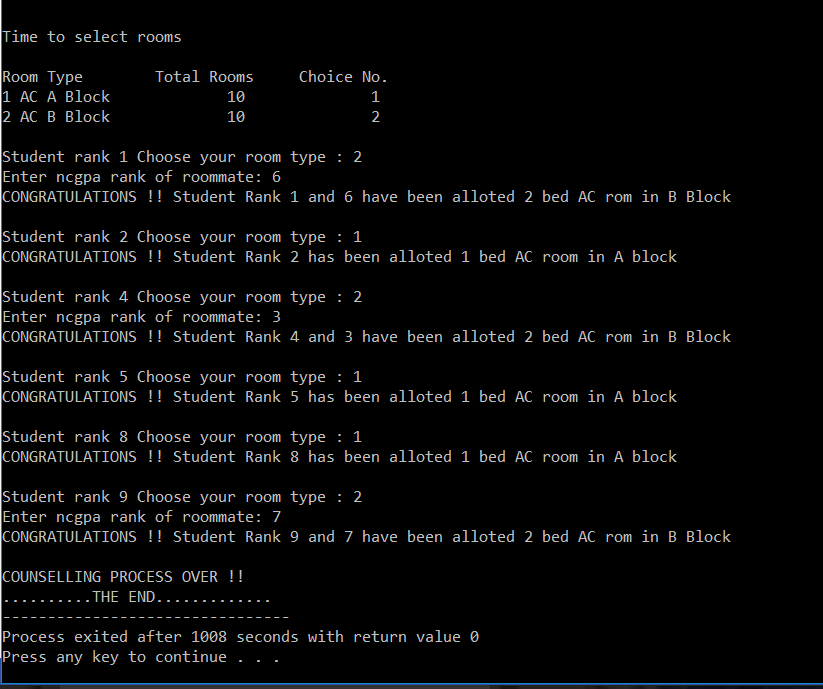


Students not attending counselling and attending counselling



Students choosing room

'



# REFERENCES

* Data structures in C by Balaguruswamy.
* tutorialspoint.com
* see-programming.blogspot.in