PROGRAMMING PRACTICE LAB

Name :- **Anubhab Rakshit**
Class :- **BCSE-II**
Sec :- **A1**
Roll Number :- **002410501029**

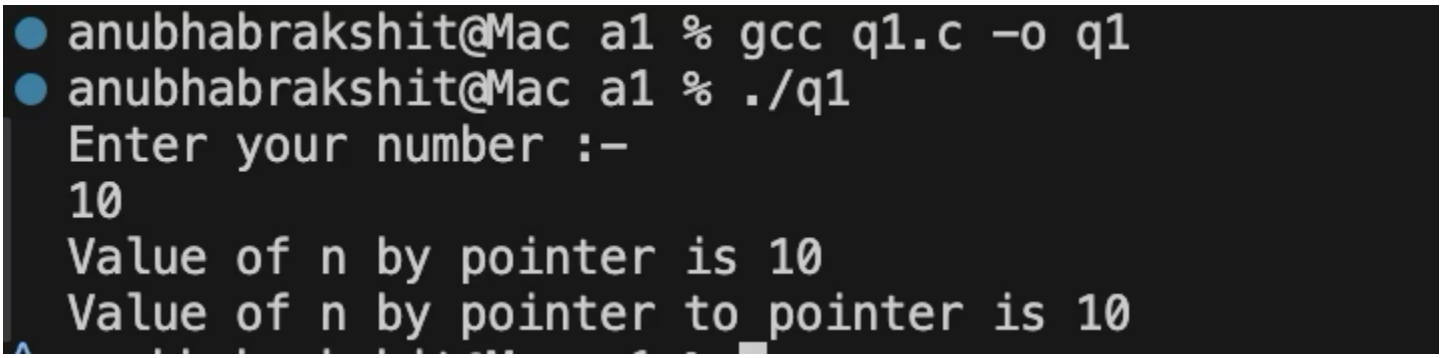
INDEX

ASSIGNMENT - 1

Q1. Write a program that will have an integer variable and a pointer (say, p) pointing to it. Also have a pointer to pointer pointing to p. Take the value for the integer variable and print it using p, and pp.

```
/* Initialize pointer --> Initialize pointer to pointer --> Enter number --> do necessary steps --> print values*/

#include <stdio.h>
#include <stdlib.h>

int main(){
   int n;
   int *p;   //pointer
   int **pp;  //pointer to pointer

   printf("Enter your number :- \n");
   scanf("%d",&n);

   p=&n; //address of variable
   pp=&p; // taking address of pointer
   printf("Value of n by pointer is %d\n",*p);
   printf("Value of n by pointer to pointer is %d\n",**pp);
   return 0;
}
```

OUTPUT :-

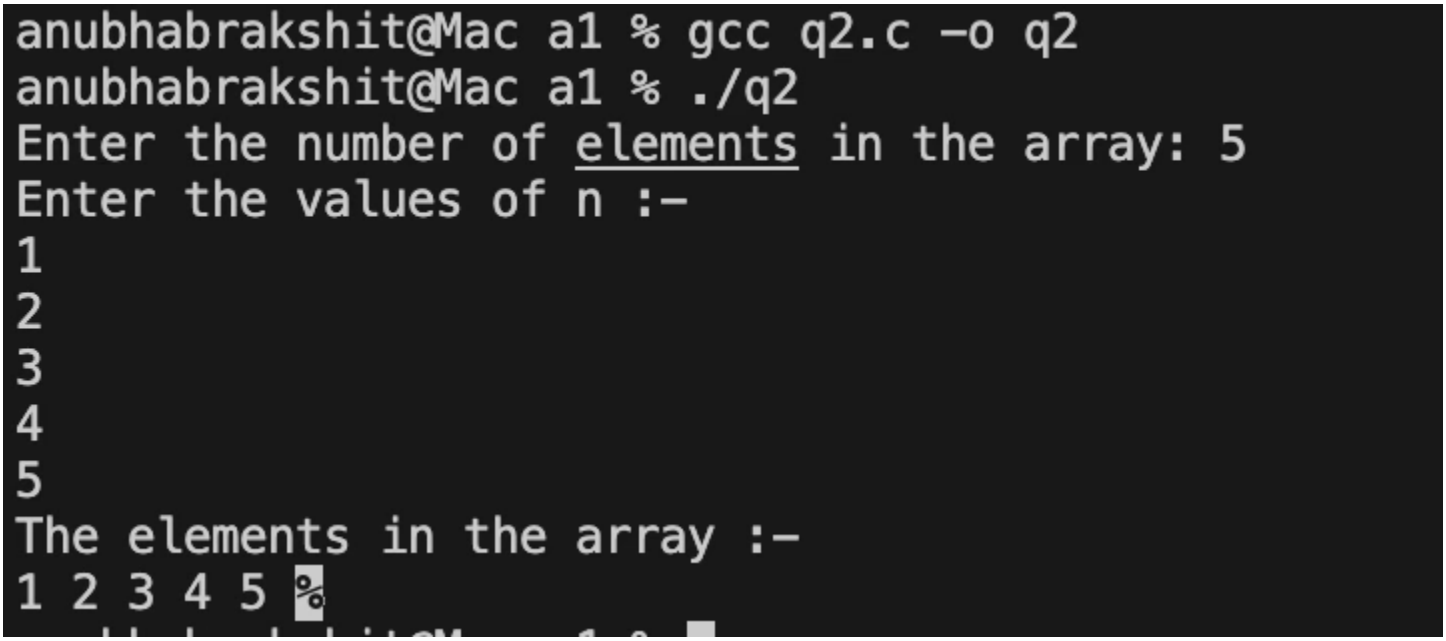


Q2. Implement a one dimensional array of integers where array size of the array will be provided during

runtime. Accept the value for the elements and print those using pointers.

```
/*
Initialize int* pointer --> Get number of elements --> Allocate memory and enter the values --> Print elements
*/


#include <stdio.h>
#include <stdlib.h>

int main(){

   int n;
   int *arr;         //array pointer
   printf("Enter the number of elements in the array: ");
   scanf("%d", &n);

   arr=(int *)malloc(sizeof(int)*n); // allocating memory for n integers

   printf("Enter the values of n :- \n");
   for(int i=0;i<n;i++){
      scanf("%d", &arr[i]); // entering n elements
   }
   printf("The elements in the array :- \n");
   for(int i=0;i<n;i++){
      printf("%d ", *(arr+i)); //printing elements of the array
   }

   return 0;

}
```

OUTPUT:-



3. Implement a two dimensional array of integers using a) array of pointers b) pointer to pointer with two malloc statements and again with one malloc statement, c) pointer to an array. Accept the value for the elements and print those.

```
#include <stdio.h>
#include <stdlib.h>

int main(){
   int ch;

   printf("Enter your choice :- \n");
   printf("\n Menu :- 1. Array of pointers \n 2. Double Malloc \n 3. Single Malloc \n 4.                    Pointer to an
Array \n");
   scanf("%d",&ch); // enter your choice for particular method

   /*
```

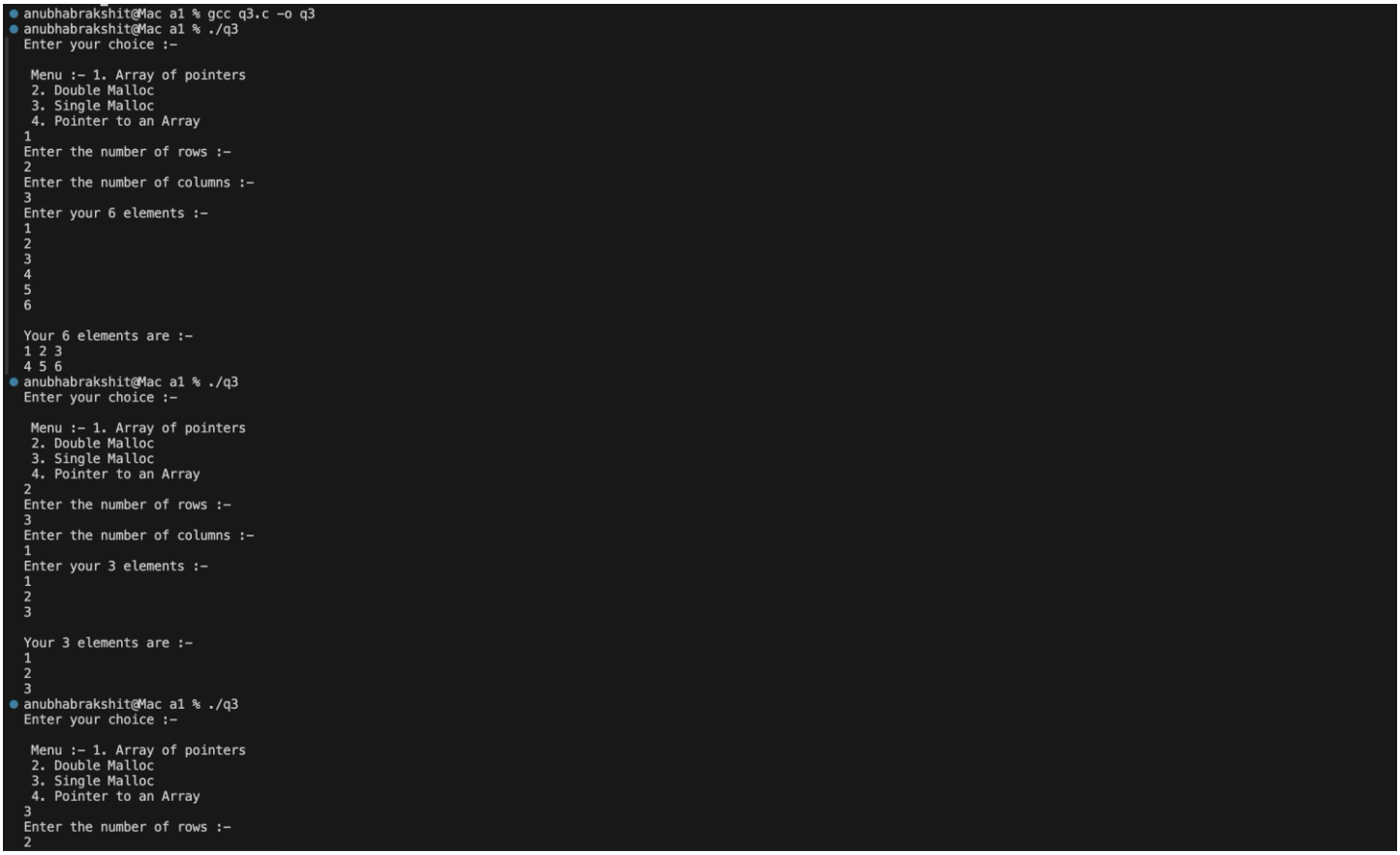
```
   Get no of rows and columns --> initialize array of pointers --> allocate memory --> enter elements --> print elements
   */

   if (ch==1){ // method-1 (array of pointers)

      printf("Enter the number of rows :- \n");
      int r;
      scanf("%d",&r); // value of row
      printf("Enter the number of columns :- \n");
      int c;
      scanf("%d",&c);  //value of column
      int* arr[r]; // array of pointers
      for(int i=0;i<r;i++){
         arr[i]=(int*)malloc(c*sizeof(int)); // allocate memory for each row
      }
      printf("Enter your %d elements :- \n" , r*c);
      for(int i=0;i<r;i++){
         for(int j=0;j<c;j++){
            scanf("%d",&arr[i][j]); // accept value for elements
         }
      }
      printf("\nYour %d elements are :- \n" , r*c);
      for(int i=0;i<r;i++){
         for(int j=0;j<c;j++){
            printf("%d ",arr[i][j]); // printing elements
         }
         printf("\n");
      }
      for(int i=0;i<r;i++){
         free(arr[i]); // free the allocated memory
      }
   }
   /*
   Get no of rows and columns --> Initialize int ** --> Allocate memory for each int* (rows) --> Enter elements --> Print elements
   */
   if (ch==2){ //method-2 (double malloc)
      printf("Enter the number of rows :- \n");
      int r;
      scanf("%d",&r); //value of row
      printf("Enter the number of columns :- \n");
      int c;
```

```
    scanf("%d",&c); //value of column
    int** arr=malloc(r*sizeof(int*)); //pointer to pointer
    for(int i=0;i<r;i++){
      arr[i]=malloc(c*sizeof(int)); //allocating memory for c integers
    }
    printf("Enter your %d elements :- \n" , r*c);
    for(int i=0;i<r;i++){
      for(int j=0;j<c;j++){
        scanf("%d",&arr[i][j]); //entering elements
      }
    }
    printf("\nYour %d elements are :- \n" , r*c);
    for(int i=0;i<r;i++){
      for(int j=0;j<c;j++){
        printf("%d ",arr[i][j]); //printing elements
      }
      printf("\n");
    }

    for(int i=0;i<r;i++){
      free (arr[i]); //free allocated memory for each row
    }
    free(arr); //free allocating memory of 2d array

  }

  /*
  Get no of rows and columns --> Allocate memory for r rows and r*c elements --> Link the address with elements
--> Enter elements --> Print elements
  */
  if (ch==3){ //method-2 (single malloc)

    printf("Enter the number of rows :- \n");
    int r;
    scanf("%d",&r);
    printf("Enter the number of columns :- \n");
    int c;
    scanf("%d",&c);

    int** arr=(int**)malloc(r*sizeof(int *)+r*c*sizeof(int));
    for(int i=0;i<r;i++){
      arr[i]=(int *)(arr+r+i*c);
    }
```

```
    printf("Enter your %d elements :- \n" , r*c);
    for(int i=0;i<r;i++){
      for(int j=0;j<c;j++){
        scanf("%d",&arr[i][j]); //entering elements
      }
    }
    printf("\nYour %d elements are :- \n" , r*c);
    for(int i=0;i<r;i++){
      for(int j=0;j<c;j++){
        printf("%d “,arr[i][j]); //printing elements
      }
      printf("\n");
    }
    free(arr); //free allocated memory
  }
  /*
  Get no of rows and columns -->  Initialize pointer to array (of column) --> Allocate memeory --> Enter elements --
> Print Elements
  */
  if (ch==4) // method-3 (pointer to an array)
  {
    printf("Enter the number of rows :- \n");
    int r;
    scanf("%d",&r);
    printf("Enter the number of columns :- \n”);
    int c;
    scanf("%d",&c);
    int (*arr)[c]; // pointer to an array
    arr=malloc(sizeof(int *)*c); // allocating memory
    printf("Enter your %d elements :- \n" , r*c);
    for(int i=0;i<r;i++){
      for(int j=0;j<c;j++){
        scanf("%d",&arr[i][j]); //entering elements
      }
    }
    printf("\nYour %d elements are :- \n" , r*c);
    for(int i=0;i<r;i++){
      for(int j=0;j<c;j++){
        printf("%d “,arr[i][j]); //printing elements
      }
      printf("\n");
    }
    free (arr); //free allocated memory
  }
```

}

OUTPUT :-


anubhabrakshit@Mac a1 % gcc q3.c -o q3
anubhabrakshit@Mac a1 % ./q3
Enter your choice :-
Menu :- 1. Array of pointers
2. Double Malloc
3. Single Malloc
4. Pointer to an Array
1
Enter the number of rows :-
2
Enter the number of columns :-
3
Enter your 6 elements :-
1
2
3
4
5
6
Your 6 elements are :-
1 2 3
4 5 6
anubhabrakshit@Mac a1 % ./q3
Enter your choice :-
Menu :- 1. Array of pointers
2. Double Malloc
3. Single Malloc
4. Pointer to an Array
2
Enter the number of rows :-
3
Enter the number of columns :-
1
Enter your 3 elements :-
1
2
3
Your 3 elements are :-
1
2
3
anubhabrakshit@Mac a1 % ./q3
Enter your choice :-
Menu :- 1. Array of pointers
2. Double Malloc
3. Single Malloc
4. Pointer to an Array
3
Enter the number of rows :-
2


Enter the number of columns :-
2
Enter your 4 elements :-
1
2
3
4
Your 4 elements are :-
1 2
3 4
anubhabrakshit@Mac a1 % ./q3
Enter your choice :-
Menu :- 1. Array of pointers
2. Double Malloc
3. Single Malloc
4. Pointer to an Array
4
Enter the number of rows :-
3
Enter the number of columns :-
2
Enter your 6 elements :-
1
2
3
4
5
6
Your 6 elements are :-
1 2
3 4
5 6

4. Implement the programs in Q.2 and 3 breaking it into functions for i) getting the dimensions from user, ii) dynamic memory allocation, iii) accepting the values and iv) printing the values.

```
#include <stdio.h>
#include <stdlib.h>

// ================= Question 2 (1D array) =================
int q2_dim(){ // function for dimension of 1d array
   printf("Enter the dimension for the 1d array :- \n");
   int num;
   scanf("%d", &num);
   return num;
}

void q2_alloc(int *arr,int n){ // function for allocating memory to 1d array
   arr = (int *)malloc(n * sizeof(int));
   if (!arr) {
      printf("Memory allocation failed!\n");
      return;
   }
   printf("Enter the elements :- \n");
   for(int i=0;i<n;i++){
      scanf("%d",arr+i);
   }
   printf("Your elements :- \n");
   for(int i=0;i<n;i++){
      printf("%d ",*(arr+i));
   }
   printf("\n");
   free(arr);
}


void get(int **arr,int r,int c){ // function for entering elements into 2d array
   printf("Enter your %d elements :- \n" , r*c);
   for(int i=0;i<r;i++){
      for(int j=0;j<c;j++){
         scanf("%d",&arr[i][j]);
      }
   }
}
```

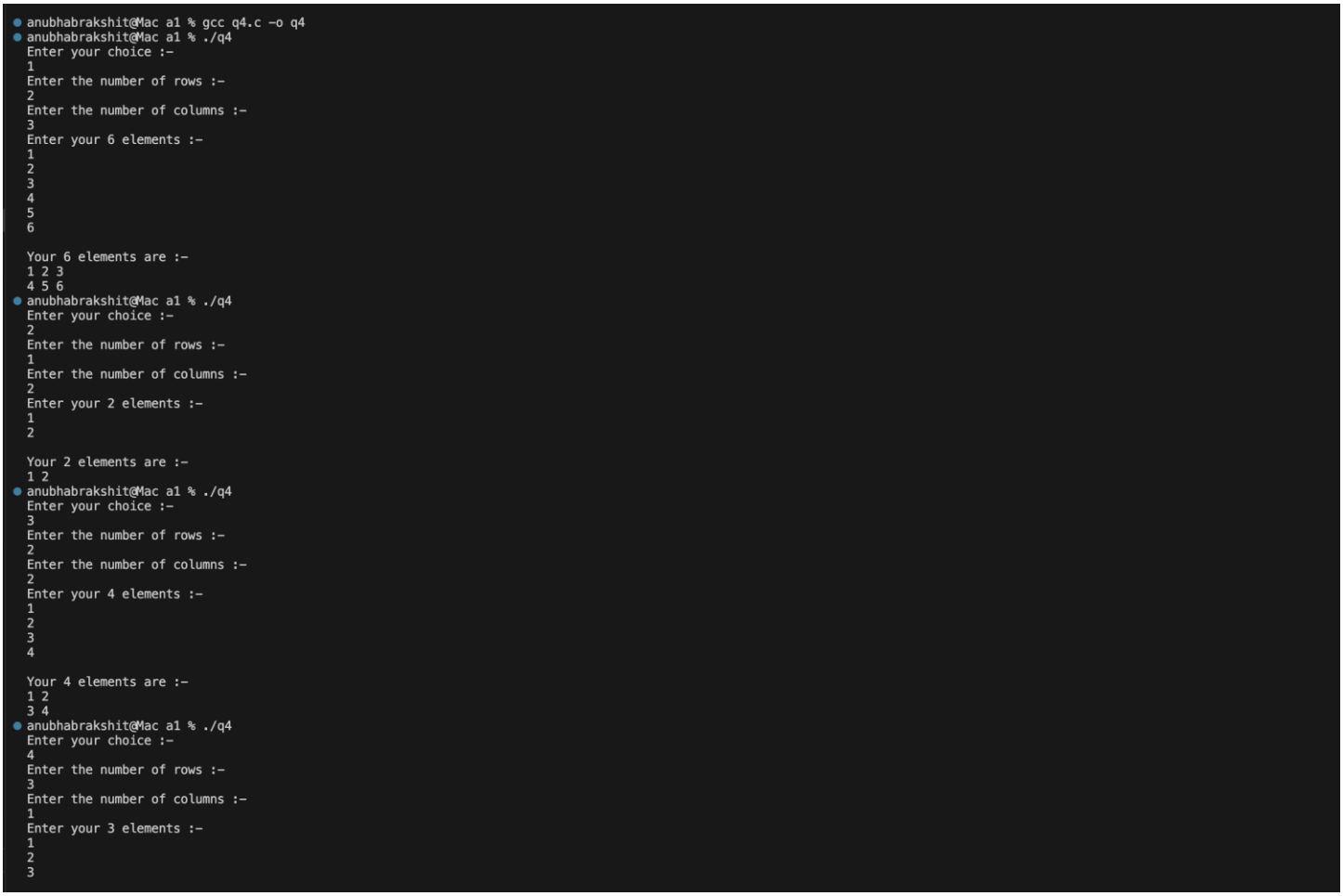
```
void print(int **arr,int r,int c){ // function for printing elements of 2d array
   printf("\nYour %d elements are :- \n" , r*c);
   for(int i=0;i<r;i++){
      for(int j=0;j<c;j++){
         printf("%d ",arr[i][j]);
      }
      printf("\n");
   }
}

void get_ptrarray(int r, int c, int arr[r][c]) {
   printf("Enter your %d elements :- \n", r*c);
   for (int i=0;i<r;i++) {
      for (int j=0;j<c;j++) {
         scanf("%d",&arr[i][j]);
      }
   }
}

void print_ptrarray(int r, int c, int arr[r][c]) {
   printf("\nYour %d elements are :- \n", r*c);
   for (int i=0;i<r;i++) {
      for (int j=0;j<c;j++) {
         printf("%d ", arr[i][j]);
      }
      printf("\n");
   }
}


// ================= Allocation methods =================
void methoda(){ // method of array of pointers (2d array)
   printf("Enter the number of rows :- \n");
   int r;
   scanf("%d",&r);
   printf("Enter the number of columns :- \n");
   int c;
   scanf("%d",&c);

   int* arr[r];
   for(int i=0;i<r;i++){
      arr[i]=(int*)malloc(c*sizeof(int));
```

```
    }
    get(arr,r,c);
    print(arr,r,c);
    for(int i=0;i<r;i++){
        free(arr[i]);
    }
}

void methodb_two(){ // method for double malloc (2d array)
    printf("Enter the number of rows :- \n");
    int r;
    scanf("%d",&r);
    printf("Enter the number of columns :- \n");
    int c;
    scanf("%d",&c);
    int** arr=malloc(r*sizeof(int*));
    for(int i=0;i<r;i++){
        arr[i]=malloc(c*sizeof(int));
    }
    get(arr,r,c);
    print(arr,r,c);
    for(int i=0;i<r;i++){
        free (arr[i]);
    }
    free(arr);
}

void methodb_single(){ // method for single malloc (1d block used as 2d)
    printf("Enter the number of rows :- \n");
    int r;
    scanf("%d",&r);
    printf("Enter the number of columns :- \n");
    int c;
    scanf("%d",&c);
    int** arr=(int**)malloc(r*sizeof(int *)+r*c*sizeof(int));
    int *ptr=(int*)(arr+r);
    for(int i=0;i<r;i++){
        arr[i]=ptr+i*c;
    }
    get(arr,r,c);
    print(arr,r,c);
    free(arr);
}
```

```
void methodc(){ // method of pointer to an array
   printf("Enter the number of rows :- \n");
   int r;
   scanf("%d",&r);
   printf("Enter the number of columns :- \n");
   int c;
   scanf("%d",&c);

   int (*arr)[c] = malloc(r * sizeof(*arr));  // allocate r rows of c integers
   if (!arr) {
      printf("Memory allocation failed!\n");
      return;
   }

   get_ptrarray(r, c, arr);
   print_ptrarray(r, c, arr);

   free(arr); // free allocated memory
}



int main(){
   int ch;
   printf("Enter your choice :- \n");
   scanf("%d",&ch); // Choice for switch case
   switch(ch){
      case 1:
         methoda(); // calling method for array of pointers
         break;
      case 2:
         methodb_two(); // calling method with 2 malloc statements
         break;
      case 3:
         methodb_single(); // calling method with 1 malloc statement
         break;
      case 4:
         methodc(); // calling method of pointer to an array
         break;
      case 5: {
         printf("For Question 2 \n");
         int n=q2_dim(); // getting dimension
         int *arr;
```

```
        q2_alloc(arr,n); // allocating memory and print elements
        break;
      }
      default:
        printf("Invalid choice\n");
   }
   return 0;
}
```

OUTPUT :-



5. Store name and age of number of persons (number provided at run time). Collect the data and display data in the ascending order of age. Implement without using structure. Write functions for memory allocation of the list, sorting and display of data.

```
/* Create datatype for saving names , ages  --> Enter details --> Pass to swap function for swapping ages (sorting in ascending order) and corresponding indexes(stored in an array) -> Print the names according to the indexes*/

#include <stdio.h>
#include <stdlib.h>



void swap_age(int *ages, char **names,int n,int *index){ // function for swaping ages
   for(int i=0;i<n-1;i++) // Bubble sort
   {
      for(int j=0;j<n-i-1;j++){
         if(ages[j]>ages[j+1]){
            int temp = ages[j]; //swapping ages (sorting in ascending order)
            ages[j] = ages[j+1];
            ages[j+1] = temp;
            temp=index[j]; // swapping indexes (sorting in ascending order)
            index[j]=index[j+1];
            index[j+1]=temp;
         }
      }
   }

}

void print_names(int *index,int n,char** names){ // function for printing names
   printf("Names in ascending order of age :- \n");
   for(int i=0;i<n;i++){
      int in=index[i];
      printf("%s\n",names[in]); // ages and indexes got swapped , so automatically names would be printed in sorted index order
   }

}
void get(){ // function for entering details
   int n;
```

```
    printf("Enter the number of persons :- \n");
    scanf("%d",&n);

    char** names=(char **)malloc(sizeof(char* )*n);
    int* ages=(int*)malloc(sizeof(int)*n);
    int *index =(int *)malloc(sizeof(int)*n);
    for(int i=0;i<n;i++){
        index[i]=i;
        names[i]=(char*)malloc(sizeof(char)*100);
        printf("Enter the name and age of person %d :- \n",i+1);
        scanf("%s %d",names[i],ages+i);
        getchar();
    }
    swap_age(ages,names,n,index); // calling swap age function
    print_names(index,n,names);
}
int main(){ // main function
    get();
}
```

OUTPUT :-

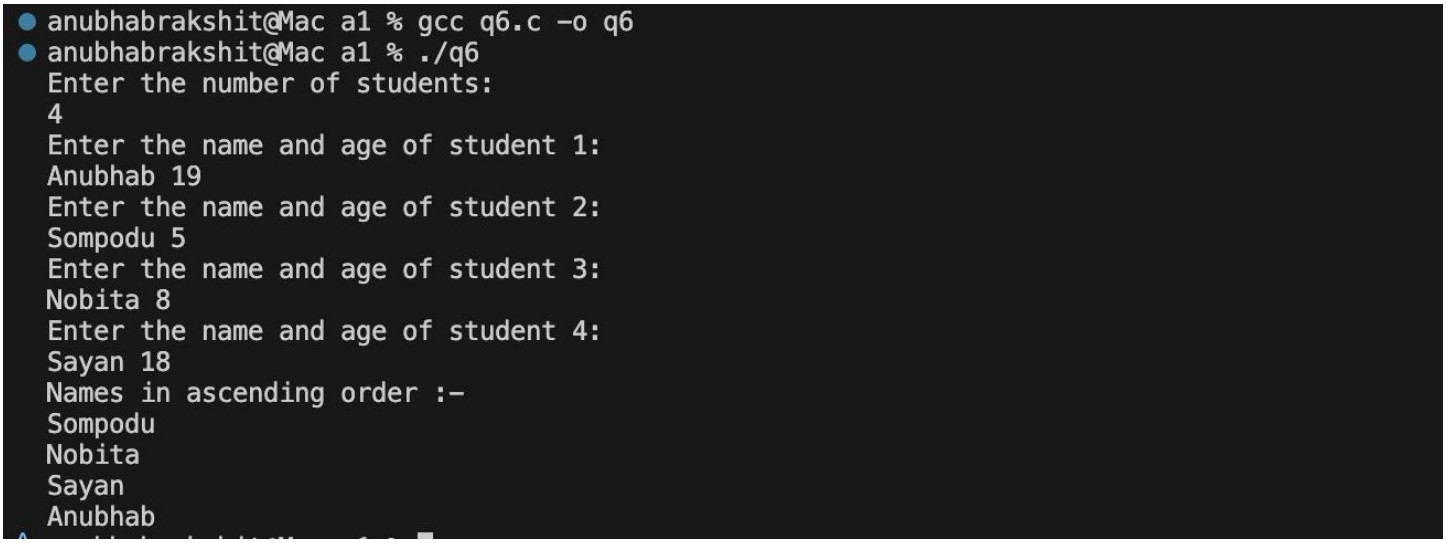
```
anubhabrakshit@Mac a1 % gcc q5.c -o q5
anubhabrakshit@Mac a1 % ./q5
Enter the number of persons :-
3
Enter the name and age of person 1 :-
Anubhab 19
Enter the name and age of person 2 :-
Sompodu 5
Enter the name and age of person 3 :-
Doraemon 18
Names in ascending order of age :-
Sompodu
Doraemon
Anubhab
```

6. Implement Q.5 using structure.

```
#include <stdlib.h>
#include <stdio.h>


typedef struct { // struct for storing data
   char name[100];
   int age;
}Student;

void swap_age(Student *arr,int n){ // function for swapping age
   for(int i=0;i<n-1;i++){ // Bubble sort
      for(int j=0;j<n-i-1;j++){
         if(arr[j].age>arr[j+1].age){ // comparing ages inside each struct (sorting in ascending order)
            Student temp=arr[j]; // swapping structs
            arr[j]=arr[j+1];
            arr[j+1]=temp;
         }
      }
   }
}

void get(){ // entering details
   int n;
   printf("Enter the number of students: \n");
   scanf("%d", &n);
   Student* arr = (Student*)malloc(n * sizeof(Student)); // allocating memeory of n structs named Student
   for (int i = 0; i < n; i++) {
      printf("Enter the name and age of student %d: \n", i + 1);
      scanf("%s %d", arr[i].name, &arr[i].age);
      getchar();
   }

   swap_age(arr,n); // calling function for sorting ages
   printf("Names in ascending order :- \n"); //printing names in ascending order
   for (int i = 0; i < n; i++) {
      printf("%s %d\n", arr[i].name, arr[i].age);
   }
}
int main(){ // main function
   get(); // calling function for getting details
```

}

OUTPUT :-



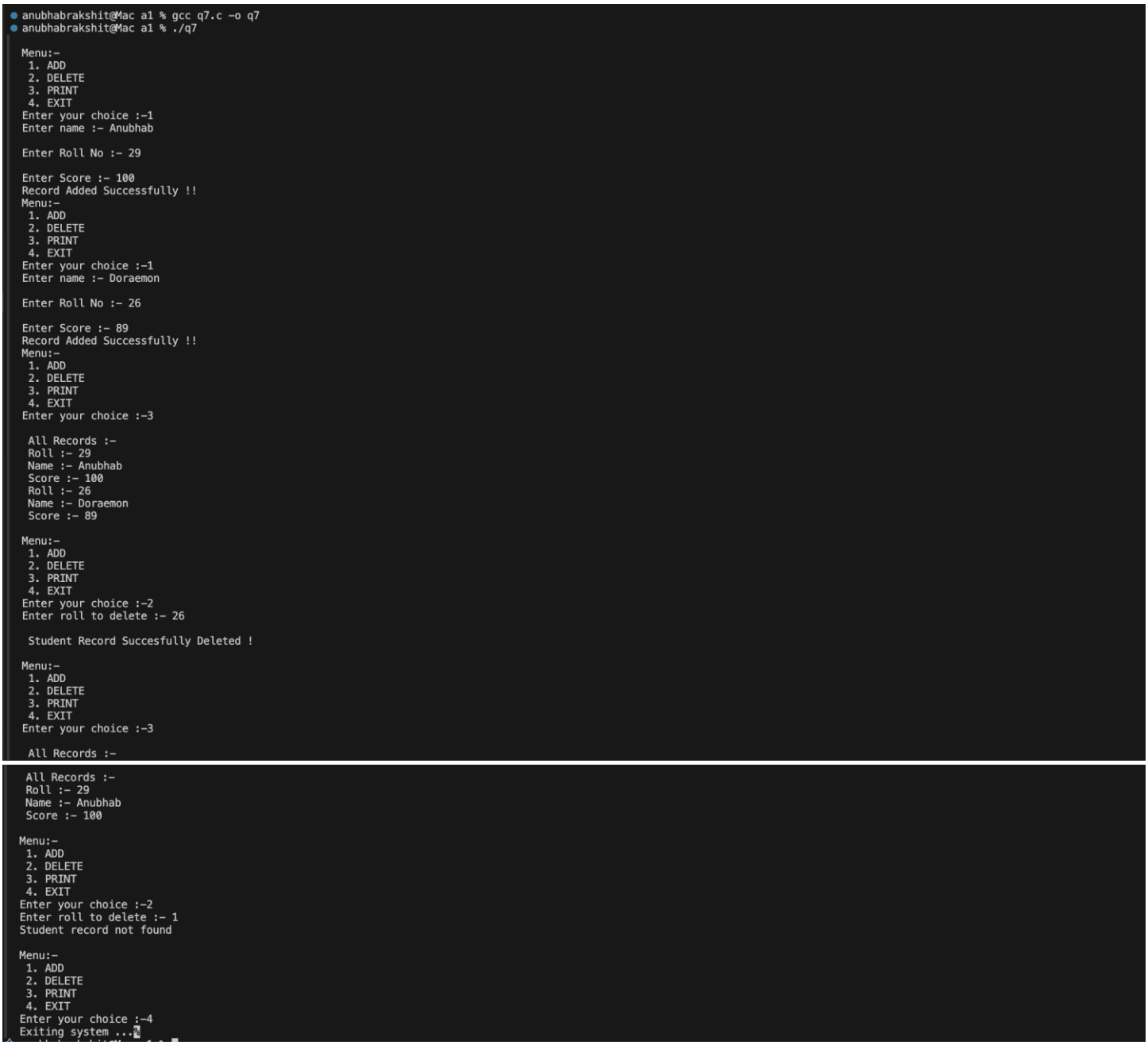
7. Maintain a list to store roll, name and score of students. As and when required student record may be added or deleted. Also, the list has to be displayed. Design suitable functions for different operations.

```
/*
Defining a student structure —> Adding students data function —> Deleting student data function (by roll)  —>
Display function —>Menu in main and call from there —> free memory and exit the system upon desired choice
*/


#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct { // defining struct
   int roll;
   char name[41];
   int score;
}Student;
```

```
Student* students=NULL; //allocating a null Student* pointer to hold all values
int count=0; // number of values

void add(int roll,char *name,int score){ // add function
   students=(Student* )realloc(students,(count+1)*sizeof(Student)); // reallocating memory (increasing to hold 1
more student)
   students[count].roll=roll;
   strcpy(students[count].name,name);
   students[count].score=score;
   count+=1;

   printf("Record Added Successfully !!");
}
void delete(int roll){ //delete function
   int f=-1;
   for(int i=0;i<count;i++){
      if(students[i].roll==roll){ //data found
         f=i;
         break;
      }
   }
   if(f==-1) //data not found
   {
      printf("Student record not found \n");
      return;
   }
   for(int i=f;i<count-1;i++){ //shifting elements to delete the record
      students[i]=students[i+1];
   }
   count-=1; //decreasing student count by 1
   students=(Student*)realloc(students,count*sizeof(Student)); //reallocating memory for new number of students
   printf("\n Student Record Succesfully Deleted ! \n");
}

void display(){ //display function
   if (count==0){
      printf("No Records found"); //no records found
      return;
   }
   printf("\n All Records :- \n");
   for(int i=0;i<count;i++){
      printf(" Roll :- %d \n Name :- %s \n Score :- %d \n",students[i].roll,students[i].name,students[i].score);
   }
```

```
}
int main()
{
   int ch,roll,score;
   char name[41];
   while(1){
      printf("\nMenu:- \n 1. ADD \n 2. DELETE \n 3. PRINT \n 4. EXIT\n"); //menu
      printf("Enter your choice :-");
      scanf("%d",&ch);
      switch(ch){
         case 1: //add data
            printf("Enter name :- ");
            getchar();
            scanf("%[^\n]s",name);
            printf("\nEnter Roll No :- ");
            scanf("%d",&roll);
            printf("\nEnter Score :- ");
            scanf("%d",&score);
            add(roll,name,score);
            break;
         case 2: //delete data
            printf("Enter roll to delete :- ");
            scanf("%d",&roll);
            delete(roll);
            break;
         case 3: //display data
            display();
            break;
         case 4: //exit system
            free(students);
            printf("Exiting system ...");
            exit(0);
         default:
            printf("Invalid choice ");
      }
   }
}
```

OUTPUT :-



```
anubhabrakshit@Mac a1 % gcc q7.c -o q7
anubhabrakshit@Mac a1 % ./q7

Menu:-
 1. ADD
 2. DELETE
 3. PRINT
 4. EXIT
Enter your choice :-1
Enter name :- Anubhab

Enter Roll No :- 29

Enter Score :- 100
Record Added Successfully !!
Menu:-
 1. ADD
 2. DELETE
 3. PRINT
 4. EXIT
Enter your choice :-1
Enter name :- Doraemon

Enter Roll No :- 26

Enter Score :- 89
Record Added Successfully !!
Menu:-
 1. ADD
 2. DELETE
 3. PRINT
 4. EXIT
Enter your choice :-3

 All Records :-
 Roll :- 29
 Name :- Anubhab
 Score :- 100
 Roll :- 26
 Name :- Doraemon
 Score :- 89

Menu:-
 1. ADD
 2. DELETE
 3. PRINT
 4. EXIT
Enter your choice :-2
Enter roll to delete :- 26

 Student Record Succesfully Deleted !

Menu:-
 1. ADD
 2. DELETE
 3. PRINT
 4. EXIT
Enter your choice :-3

 All Records :-
 Roll :- 29
 Name :- Anubhab
 Score :- 100

Menu:-
 1. ADD
 2. DELETE
 3. PRINT
 4. EXIT
Enter your choice :-2
Enter roll to delete :- 1
Student record not found

Menu:-
 1. ADD
 2. DELETE
 3. PRINT
 4. EXIT
Enter your choice :-4
Exiting system ...
```

8.Consider an array that stores roll, name, and score of number of students. Develop a function to sort the array. User of sort() will develop the comparison function for sorting on roll/score and ascending or descending order and reuse the same sort() function.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>

// structure to hold student details
typedef struct {
   int roll;
   char name[100];
   int score;
} Student;

// generic sort function using function pointer
void sort(Student *arr, int n, bool(*fp)(Student, Student)) {
   for (int i = 0; i < n - 1; i++) {
      for (int j = i+1; j < n; j++) {
         if (fp(arr[i], arr[j])) {
            Student temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
         }
      }
   }
}

// comparison functions
bool rollAsc(Student a, Student b)   { return a.roll > b.roll; } //logic defination
bool rollDesc(Student a, Student b)  { return b.roll > a.roll; }
bool scoreAsc(Student a, Student b)  { return a.score > b.score; }
bool scoreDesc(Student a, Student b) { return b.score > a.score; }

// print student array
void display(Student *arr,int n) {
   printf("\nStudent Records\n");
   for (int i = 0; i<n; i++) {
      printf("Roll: %d, Name: %s, Score: %d\n",arr[i].roll, arr[i].name, arr[i].score);
   }
}
```

```
int main() {
   int n;
   printf("Enter number of students: ");
   scanf("%d", &n);

   Student *arr = (Student*)malloc(n * sizeof(Student));

   // input student details
   for (int i = 0; i < n; i++) {
      printf("Enter roll: ");
      scanf("%d", &arr[i].roll);
      printf("Enter name: ");
      getchar(); // clear newline
      scanf("%[^\n]s", arr[i].name);
      printf("Enter score: ");
      scanf("%d", &arr[i].score);
   }

   // menu loop
   while(1){
      int ch;
      printf("\nSort by :- \n1. Roll-Ascending \n2. Roll-Descending \n3. Score-Ascending \n4. Score-
Descending\n");
      scanf("%d", &ch);

      switch (ch){
         case 1: sort(arr,n,rollAsc); break; // call roll ascending
         case 2: sort(arr,n,rollDesc); break; // call roll descending
         case 3: sort(arr,n,scoreAsc); break; // call score ascending
         case 4: sort(arr,n,scoreDesc); break; // call score descending
         default:
            printf("Invalid choice\n");
            free(arr);
            exit(0);
      }
      display(arr, n);
   }

   free(arr); // cleanup (unreachable but good practice)
   return 0;
}
```

OUTPUT :-

```
anubhabrakshit@Mac a1 % gcc q8.c -o q8
anubhabrakshit@Mac a1 % ./q8
Enter number of students: 3
Enter roll: 29
Enter name: Anubhab
Enter score: 99
Enter roll: 26
Enter name: Sompodu
Enter score: 89
Enter roll: 18
Enter name: Virat
Enter score: 98

Sort by :-
1. Roll-Ascending
2. Roll-Descending
3. Score-Ascending
4. Score-Descending
1

Student Records
Roll: 18, Name: Virat, Score: 98
Roll: 26, Name: Sompodu, Score: 89
Roll: 29, Name: Anubhab, Score: 99

Sort by :-
1. Roll-Ascending
2. Roll-Descending
3. Score-Ascending
4. Score-Descending
2

Student Records
Roll: 29, Name: Anubhab, Score: 99
Roll: 26, Name: Sompodu, Score: 89
Roll: 18, Name: Virat, Score: 98

Sort by :-
1. Roll-Ascending
2. Roll-Descending
3. Score-Ascending
4. Score-Descending
3

Student Records
Roll: 26, Name: Sompodu, Score: 89
Roll: 18, Name: Virat, Score: 98
Roll: 29, Name: Anubhab, Score: 99

Sort by :-
1. Roll-Ascending
2. Roll-Descending
3. Score-Ascending
4. Score-Descending
4

Sort by :-
1. Roll-Ascending
2. Roll-Descending
3. Score-Ascending
4. Score-Descending
4

Student Records
Roll: 29, Name: Anubhab, Score: 99
Roll: 18, Name: Virat, Score: 98
Roll: 26, Name: Sompodu, Score: 89

Sort by :-
1. Roll-Ascending
2. Roll-Descending
3. Score-Ascending
4. Score-Descending
^C
```

ASSIGNMENT - 2

1. Write a program to store student information in a file and to do the following operations. Information includes roll, name, and score in five subjects. Use may like to add record (check if roll number is unique), display all records showing roll, name and total score. User may search for the record against a roll.
User may edit the details in a record.User may delete record. Deletion may be logical (by some means indicate it is an invalid record and to be avoided in processing) and physical(file will not have the record).

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>


//defining student structure
typedef struct {
   int roll;
   char name[41];
   int marks[5];
   int deleted; // 0 = active, 1 = logically deleted
} Student;


bool isRollUnique(char* ,int );

void add(char *filename) {

   Student s;
   //opening file in append and binary mode
   FILE *fp = fopen(filename, "ab+");
   if(fp==NULL){
    printf("Error opening file!\n");
    return;
   }

   //input
   printf("Enter roll no :- ");
   scanf("%d",&s.roll);
```

```
    //checking for unique roll number
    if(!isRollUnique(filename,s.roll)){
      printf("Roll number already exists!\n");
      fclose(fp);
      return;
    }

    //input  name and marks
    printf("Enter name :- ");
    getchar();

    scanf("%[^\n]s",s.name);

    printf("Enter marks in 5 subjects :- \n");

    for (int i=0;i<5;i++) {
       scanf("%d", &s.marks[i]);
    }

    //setting variable active
    s.deleted = 0;
    fwrite(&s,sizeof(Student),1,fp);
    printf("Record added\n");

    fclose(fp);
}

void display(char *filename) { // display function
    FILE *fp = fopen(filename, "rb");
    if (fp==NULL) {

       printf("No file found!\n");
       return;
    }

    Student s;
    printf("\nDetails in file :- \n");
    while (fread(&s, sizeof(Student), 1, fp)) {
       if (!s.deleted) { // not temporarily deleted
          int total = 0;
          for (int i = 0; i < 5; i++) {
             total += s.marks[i]; // summation of marks
```

```
            }
            printf("Name :- %s\n", s.name);
            printf("Roll :- %d\n", s.roll);
            printf("Total :- %d\n\n", total);
        }
    }

    fclose(fp);
}

void search(char *filename) { // search function
    int roll,f=0;



    FILE *fp = fopen(filename, "rb");
    if (!fp) {
        printf("File doesn't exist!\n");
        return;
    }

    printf("Enter roll to search: ");
    scanf("%d", &roll);

    Student s;
    while(fread(&s, sizeof(Student), 1, fp)){ //reading till end of file

        if (s.roll==roll && !s.deleted) { //roll matched and not temporarily deleted

            printf("Record Found!\n  Roll :- %d\nName :- %s\nMarks :- ", s.roll, s.name);
            for (int i=0;i<5;i++){
                printf("%d ",s.marks[i]);
            }
            printf("\n");
            f=1;
            break;
        }
    }

    if (f==0) {
        printf("Record not found!\n");
    }
```

```
    fclose(fp);

}

void edit(char *filename){ //edit data in function

    int roll, f=0;
    FILE *fp = fopen(filename, "rb+");
    if (fp==NULL) {
        printf("File not found!\n");
        return;
    }



    printf("Enter roll to edit: ");

    scanf("%d", &roll);

    Student s;
    while (fread(&s, sizeof(Student), 1, fp)) {

        if (s.roll == roll && !s.deleted) { //data to be edited (found)
            f=1;
            printf("Enter new name :- ");
            getchar();
            scanf("%[^\n]s",s.name);

            printf("Enter new marks for all subjects :- ");

            for (int i=0;i<5;i++)
            {
                scanf("%d", &s.marks[i]);
            }
            fseek(fp, -sizeof(Student), SEEK_CUR); // moving  cursor back with sizeof(Student)
            fwrite(&s,sizeof(Student),1,fp); // writing new data
            printf("Record updated\n");
            break;
        }
    }

    if (f==0)
    {
```

```
        printf("Record not found!\n");
    }
    fclose(fp);
}

void logicalDelete(char *filename) { //logical deletion function (stays in file but is deleted)

    int roll, f= 0;
    FILE *fp = fopen(filename, "rb+");
    if (fp==NULL) {
        printf("File not found!\n");
        return;
    }



    printf("Enter roll to logically delete :- ");
    scanf("%d", &roll);



    Student s;
    while (fread(&s,sizeof(Student),1,fp)) {
        if (s.roll==roll && !s.deleted) {
            s.deleted = 1; // changing from 0 to 1 , here 0 is not logically deleted and 1 is logically deleted
            fseek(fp, -sizeof(Student), SEEK_CUR); // moving cursor back by 1 * sizeof(Student)
            fwrite(&s,sizeof(Student),1,fp); // writing new data
            printf("Record logically deleted\n");
            f=1;
            break;
        }
    }

    if(f==0)
    {
        printf("Record not found\n");
    }
    fclose(fp);
}
void physicalDelete(char *filename) { // physical deleting all data which are logically deleted

    FILE *fp = fopen(filename, "rb");

    FILE *temp = fopen("temp.dat", "wb"); // temporarily creating a new file
```

```
    if (fp==NULL||temp==NULL) {
        printf("Error \n");
        return;
    }
    //manually only storing elements that are not logically deleted
    Student s;
    while (fread(&s,sizeof(Student),1,fp)) {
        if(!s.deleted)
        {
            fwrite(&s,sizeof(Student),1,temp);
        }
    }

    fclose(fp);
    fclose(temp);
    remove(filename); //removing original file
    rename("temp.dat", filename); //renaming temp file to original file
    printf("Physical deletion completed!\n");
}

bool isRollUnique(char* filename,int roll){

    FILE *fp = fopen(filename, "rb");
    if (fp==NULL)
    {
        return 1; // no file means unique
    }
    Student s;
    while (fread(&s,sizeof(Student),1,fp)) {
      if (s.roll==roll &&!s.deleted) {
            fclose(fp);
            return false;
        }
    }

    fclose(fp);
    return true;
}

int main(int argc,char *argv[]) { // passing filename as command line arguments
    int ch;
    while (1) {
```

```
    //final menu for operations
    printf("\nMenu:\n");
     printf("1.Add Record\n");
     printf("2.Display Records\n");
     printf("3.Search by Roll\n");
     printf("4.Edit\n");
     printf("5.Logical Delete\n");
     printf("6.Physical Delete\n");
     printf("7.Exit\n");
     printf("Enter: ");
     scanf("%d", &ch);

     switch (ch) {
        case 1:
           add(argv[1]);  // call add data function

           break;
        case 2:
           display(argv[1]);  // call display data function
           break;
        case 3:
           search(argv[1]);  // call search data function (search data by roll)
           break;
        case 4:
           edit(argv[1]); // call edit data function (edit data by roll)
           break;
        case 5:
           logicalDelete(argv[1]);  // call logical delete function (delete by roll)
           break;
        case 6:
           physicalDelete(argv[1]); // call physical delete function (removes from file)
           break;
        case 7:
           exit(0); // exiting the system
        default: printf("Invalid choice!\n");

     }
  }
  return 0;
}
```

OUTPUT :-

```
anubhabrakshit@Mac a2 % gcc q1.c -o q1
anubhabrakshit@Mac a2 % ./q1 students

Menu:
1.Add Record
2.Display Records
3.Search by Roll
4.Edit
5.Logical Delete
6.Physical Delete
7.Exit
Enter: 1
Enter roll no :- 29
Enter name :- Anubhab
Enter marks in 5 subjects :-
100 98 96 94 92
Record added

Menu:
1.Add Record
2.Display Records
3.Search by Roll
4.Edit
5.Logical Delete
6.Physical Delete
7.Exit
Enter: 1
Enter roll no :- 26
Enter name :- Doraemon
Enter marks in 5 subjects :-
80 82 84 86 88
Record added

Menu:
1.Add Record
2.Display Records
3.Search by Roll
4.Edit
5.Logical Delete
6.Physical Delete
7.Exit
Enter: 2

Details in file :-
Name :- Anubhab
Roll :- 29
Total :- 480

Name :- Doraemon
Roll :- 26
Total :- 420


Menu:
1.Add Record
2.Display Records
3.Search by Roll
4.Edit
```

```
Menu:
1.Add Record
2.Display Records
3.Search by Roll
4.Edit
5.Logical Delete
6.Physical Delete
7.Exit
Enter: 3
Enter roll to search: 29
Record Found!
  Roll :- 29
Name :- Anubhab
Marks :- 100 98 96 94 92

Menu:
1.Add Record
2.Display Records
3.Search by Roll
4.Edit
5.Logical Delete
6.Physical Delete
7.Exit
Enter: 4
Enter roll to edit: 26
Enter new name :- Nobita
Enter new marks for all subjects :- 88 86 90 92 94
Record updated

Menu:
1.Add Record
2.Display Records
3.Search by Roll
4.Edit
5.Logical Delete
6.Physical Delete
7.Exit
Enter: 2

Details in file :-
Name :- Anubhab
Roll :- 29
Total :- 480

Name :- Nobita
Roll :- 26
Total :- 450


Menu:
1.Add Record
2.Display Records
3.Search by Roll
4.Edit
5.Logical Delete
6.Physical Delete
7.Exit
Enter: 5
Enter roll to logically delete :- 26
```

```
Record logically deleted

Menu:
1.Add Record
2.Display Records
3.Search by Roll
4.Edit
5.Logical Delete
6.Physical Delete
7.Exit
Enter: 2

Details in file :-
Name :- Anubhab
Roll :- 29
Total :- 480


Menu:
1.Add Record
2.Display Records
3.Search by Roll
4.Edit
5.Logical Delete
6.Physical Delete
7.Exit
Enter: 6
Physical deletion completed!

Menu:
1.Add Record
2.Display Records
3.Search by Roll
4.Edit
5.Logical Delete
6.Physical Delete
7.Exit
Enter: 7
```

THESE PROGRAMS HAS BEEN CHECKED TILL THE LAST LAB CLASS