

LAB REPORT

**CSE2011 – DATA STRUCTURES AND ALGORITHMS LAB**



**(B.Tech. CSE Specialisation in Bioinformatics)**

**WINTER SEMESTER 2020-2021**

|  |  |
| --- | --- |
| **Name:** | ALOK MATHUR |
| **Reg. No:** | 20BCB0086 |
| **Slot:** | L51+L52 |
| **Faculty Name:** | SRIVANI A Ma’am |

**VIT – A Place to Learn; A Chance to Grow**

**QUESTIONS, CODE && OUTPUT**

1 . Write a menu driven program to perform following functions in a doubly linked list.

i) Insertion in the beginning of the list

ii) Insertion at the end of the list

iii) Insertion in a particular location of the list

iv) Deletion based on a particular value

v) Display the contents of the list

***CODE***

#include <stdio.h>

#include <iostream>

#include <string.h>

using namespace std;

void display();

struct node{

    int data;

    struct node \*next;

    struct node \*prev;

}\*head,\*temp,\*newnode;

void insert\_at\_beg()

{

    int num;

    newnode=new struct node;*//DMA in cpp*

    cout<<"Enter the number to be inserted"<<endl;

    cin>>num;

    newnode->data=num;

    if(head==NULL)

    {

        newnode->next=NULL;

        newnode->prev=NULL;

        head=temp=newnode;

    }

    else

    {

        temp=head;

        newnode->next=temp;

        newnode->prev=NULL;

        head=newnode;

    }

}

void insert\_at\_end()

{

    int num;

    newnode=new struct node;

    cout<<"Enter the number to be inserted"<<endl;

    cin>>num;

    newnode->data=num;

    temp=head;

    while(temp->next!=NULL)

    {

        temp=temp->next;

    }

    newnode->next=NULL;

    temp->next=newnode;

    temp->next->prev=temp;

}

void insert\_at\_pos()

{

    int count=0,pos,num;

    newnode=new struct node;

    cout<<"Enter the position to be inserted"<<endl;

    cin>>pos;

    cout<<"Enter the number to be inserted"<<endl;

    cin>>num;

    newnode->data=num;

    temp=head;

    while(count!=pos-1)

    {

        temp=temp->next;

        count++;

    }

    newnode->next=temp->next;

    newnode->prev=temp;

    temp->next=newnode;

}

void delete\_at\_beg()

{

    temp=head;

    head=temp->next;

    delete(temp);

    display();

}

void delete\_at\_end()

{

    temp=head;

    while(temp->next!=NULL)

    {

        temp=temp->next;

    }

    temp->prev->next=NULL;

    delete(temp);

    display();

}

void delete\_at\_pos()

{

    int pos,count,i;

    temp=head;

    cout<<"Enter the position to be deleted"<<endl;

    cin>>pos;

    for(i=0; i<pos && head!=NULL; i++)

    {

        temp = temp->next;

    }

    temp->prev->next = temp->next;*// Assign the next pointer of node to be deleted to its previous node's prev pointer*

    temp->next->prev = temp->prev;*// Assign the prev pointer of the node to be deleted to its next node's next pointer*

    free(temp);

    display();

}

void delete\_at\_val()

{

    int val;

    struct node \*ptr;

    cout<<"Enter the value to be delted"<<endl;

    cin>>val;

    temp=head;

    while(temp->data!=val)

    {

        temp=temp->next;

    }

    ptr = temp -> next;

    temp -> next = ptr -> next;

    ptr -> next -> prev = temp;

    delete(ptr);

    display();

}

void display()

{

    struct node\* ptr;

    ptr = head;

    while(ptr != NULL) {

    cout<< ptr->data <<" ";

    ptr = ptr->next;

   }

   cout<<endl;

}

int main()

{

    cout<<"\*\*\*\*\*Doubly Linked List\*\*\*\*\*"<<endl;

    int choice;

    int head=0;

    while(choice!=9)

    {

        cout<<"The operations that can be done"<<endl;

        cout<<"1.Insert At Begining\n2.Insert At End\n3.Insert At any Position\n4.Deletion at Begining\n5.Deletion at End\n6.Deletion at a Position\n7.Deletion based on Value\n8.Display\n9.Exit"<<endl;

        cout<<"Please enter your choice"<<endl;

        cin>>choice;

        switch (choice)

        {

        case 1:

            insert\_at\_beg();

            break;

        case 2:

            insert\_at\_end();

            break;

        case 3:

            insert\_at\_pos();

            break;

        case 4:

            delete\_at\_beg();

            break;

        case 5:

            delete\_at\_end();

            break;

        case 6:

            delete\_at\_pos();

            break;

        case 7:

            delete\_at\_val();

            break;

        case 8:

            display();

            break;

        case 9:

            cout<<"Exiting"<<endl;

            break;

        default:

            cout<<"Invalid Input"<<endl;

            break;

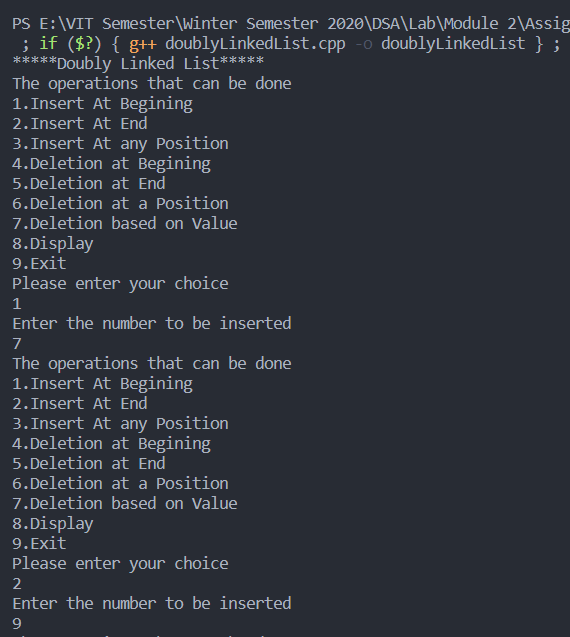
        }

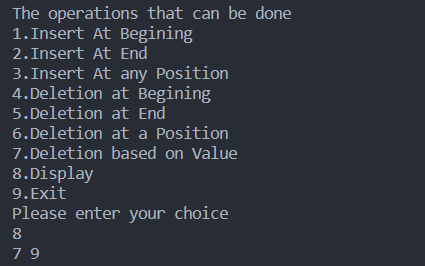
    }

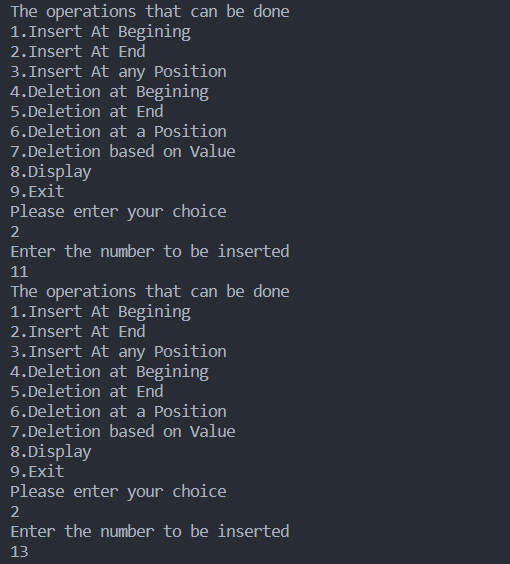
    return 1;

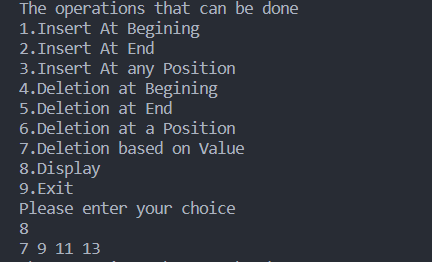
}

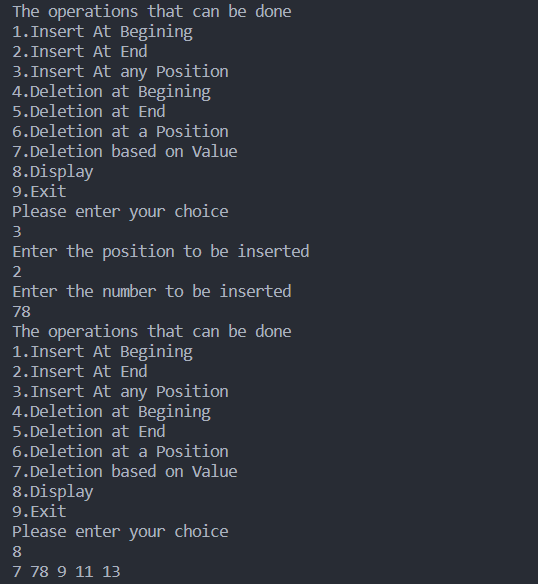
***OUTPUT***

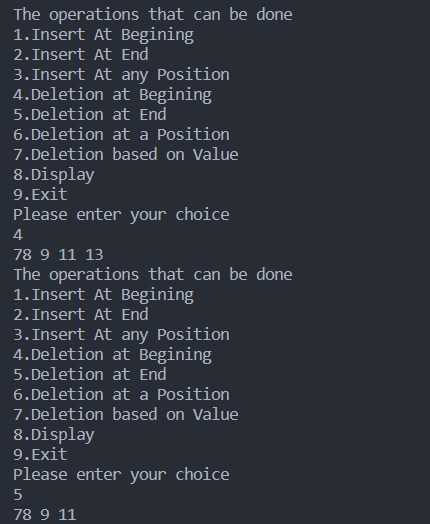












2. Write a menu driven program to perform following functions in a circularly singly linked list.

i) Insertion in the beginning of the list

ii) Insertion at the end of the list

iii) Deletion from the beginning of the list

iv) Deletion from the end of the list.

***CODE***

#include <stdio.h>

#include <iostream>

#include <string.h>

using namespace std;

void display();

struct node{

    int data;

    struct node \*next;

}\*temp,\*head,\*newnode;

void insert\_at\_beg()

{

    int num;

    newnode=new struct node;

    cout<<"Enter the number to be inserted"<<endl;

    cin>>num;

    newnode->data=num;

    if(head==NULL)

    {

        head=temp=newnode;

        temp->next=head;

    }

    else

    {

        temp=head;

        newnode->next=temp;

        head=newnode;

    }

}

void insert\_at\_end()

{

    int num;

    newnode=new struct node;

    cout<<"Enter the number to be inserted"<<endl;

    cin>>num;

    newnode->data=num;

    temp=head;

    while(temp->next!=head)

    {

        temp=temp->next;

    }

    newnode->next=head;

    temp->next=newnode;

}

void del\_at\_begin()

{

     struct node \*ptr;

    if(head == NULL)

    {

        printf("\nUNDERFLOW");

    }

    else if(head->next == head)

    {

        head = NULL;

        free(head);

        printf("\nnode deleted\n");

    }

    else

    {   ptr = head;

        while(ptr -> next != head)

            ptr = ptr -> next;

        ptr->next = head->next;

        free(head);

        head = ptr->next;

        printf("\nnode deleted\n");

    }

    display();

}

void del\_at\_end()

{

    struct node \*ptr, \*preptr;

    if(head==NULL)

    {

        printf("\nUNDERFLOW");

    }

    else if (head ->next == head)

    {

        head = NULL;

        free(head);

        printf("\nnode deleted\n");

    }

    else

    {

        ptr = head;

        while(ptr ->next != head)

        {

            preptr=ptr;

            ptr = ptr->next;

        }

        preptr->next = ptr -> next;

        free(ptr);

        printf("\nnode deleted\n");

    }

    display();

}

void display()

{

      struct node \*ptr;

    ptr=head;

    if(head == NULL)

    {

        printf("\nnothing to print");

    }

    else

    {

        printf("Printing values ... \n");

        while(ptr -> next != head)

        {

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

            ptr = ptr -> next;

        }

        printf("%d ", ptr -> data);

    }

}

int main()

{

    int choice;

    head=NULL;

    cout<<"\*\*\*Circularly Singly Linked List\*\*\*"<<endl;

    while(choice!=6)

    {

        cout<<"Please choose one of the following options"<<endl;

        cout<<"1.Insert at beginning\n2.Insert at end\n3.Delete at beginning\n4.Delete at End\n5.Display\n6.Exit"<<endl;

        cin>>choice;

        switch (choice)

        {

        case 1:

            insert\_at\_beg();

            break;

        case 2:

            insert\_at\_end();

            break;

        case 5:

            display();

            break;

        case 6:

            cout<<"Exiting..."<<endl;

        default:

            cout<<"Invalid Input"<<endl;

            break;

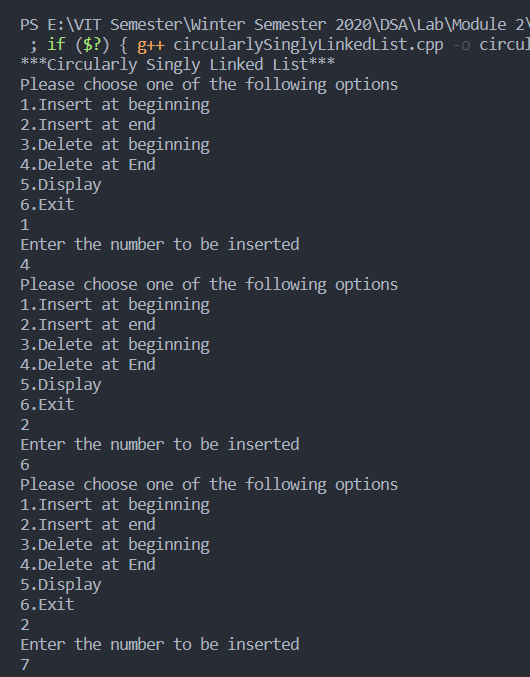
        }

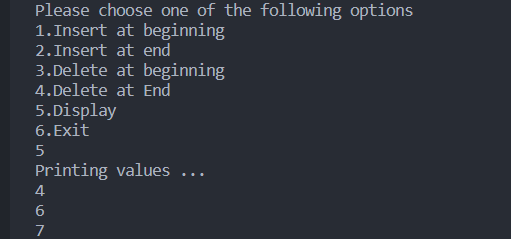
    }

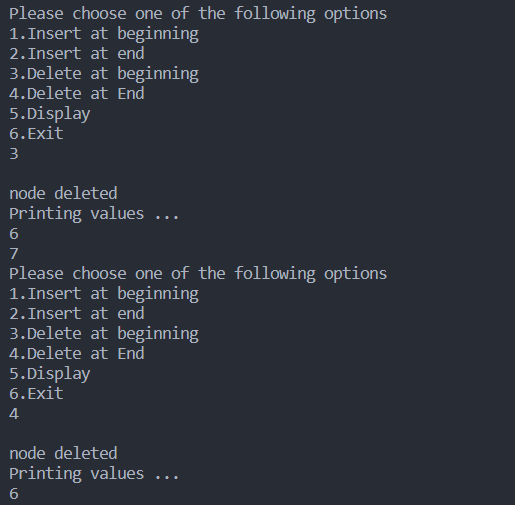
    return 1;

}

***OUTPUT***







3. Create linked list to enroll the students who wish to participate for a gaming

event by taking details like Name, Register No., Age, Phone number. Ensure that no more

than five members are there in the list with same age. Perform insertion(), deletion() and

display() operations on the Linked List.

***CODE***

#include <stdio.h>

#include <iostream>

#include <string.h>

using namespace std;

struct node{

    char name[50];

    int reg\_no;

    int age;

    int phone;

    struct node \*next;

}\*head,\*temp,\*newnode;

int counter(int );

void insert()

{

    newnode= new struct node;

    cout<<"Enter the name of participant : ";

    cin>>newnode->name;

    cout<<"\n";

    cout<<"Enter the reg no of participant : ";

    cin>>newnode->reg\_no;

    cout<<"\n";

    cout<<"Enter ur age : ";

    cin>>newnode->age;

    int c=counter(newnode->age);

    cout<<"\n";

    cout<<"Enter ur phone number : ";

    cin>>newnode->phone;

    if(head==NULL)

    {

        newnode->next=NULL;

        temp=head=newnode;

    }

    else

    {

        if(c<=4)

        {

            temp=head;

            while(temp->next!=NULL)

            {

                temp=temp->next;

            }

            newnode->next=NULL;

            temp->next=newnode;

        }

        else

        {

            cout<<"More than 5 members of same age.Sorry!"<<endl;

        }

    }

}

void del()

{

    if (head == NULL)

    {

        printf("Underflow");

    }

    else

    {

        temp=head;

        head = head->next;

        delete(temp);

    }

}

void display()

{

    struct node \*r;

    int c = 0;

    r = head;

    while (r != NULL)

    {

        c++;

        cout<<r->name<<" "<<r->age<<" "<<r->reg\_no<<" "<<r->phone;

        r = r->next;

    }

    printf("\n");

}

int counter(int *a*)

{

    struct node \*s;

    s=head;

    int count=0;

    while(s!=NULL)

    {

        if(s->age==*a*)

        {

            count++;

        }

        else

        {

            continue;

        }

        s=s->next;

    }

    return count;

}

int main()

{

    int choice;

    cout<<"\*\*Game Event\*\*"<<endl;

    head=0;

    while(choice!=4)

    {

        cout<<"Pls enter your choice"<<endl;

        cout<<"1.Register for the event\n2.Delete\n3.Display all participants\n4.Exit"<<endl;

        cin>>choice;

        switch (choice)

        {

        case 1:

            insert();

            break;

        case 2:

            del();

            break;

        case 3:

            display();

            break;

        case 4:

            cout<<"Exiting......"<<endl;

            break;

        default:

            break;

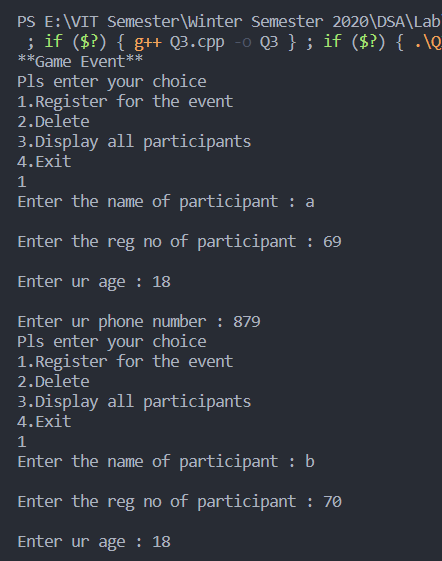
        }

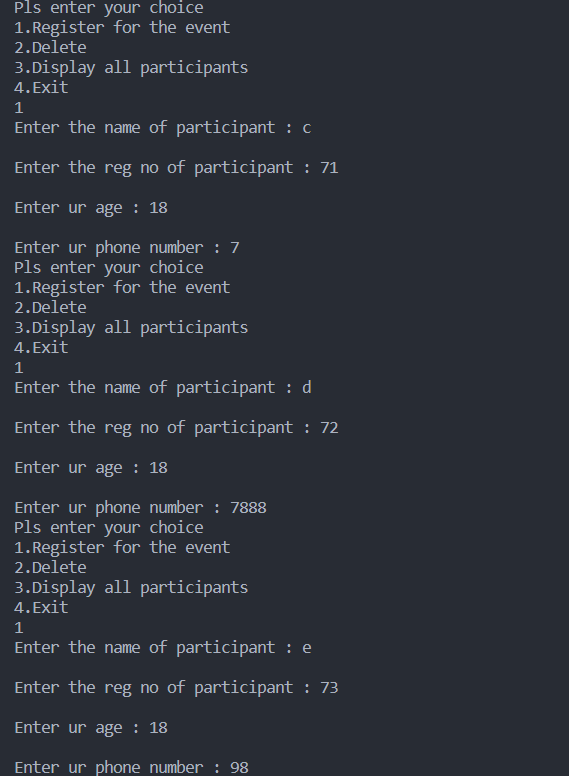
    }

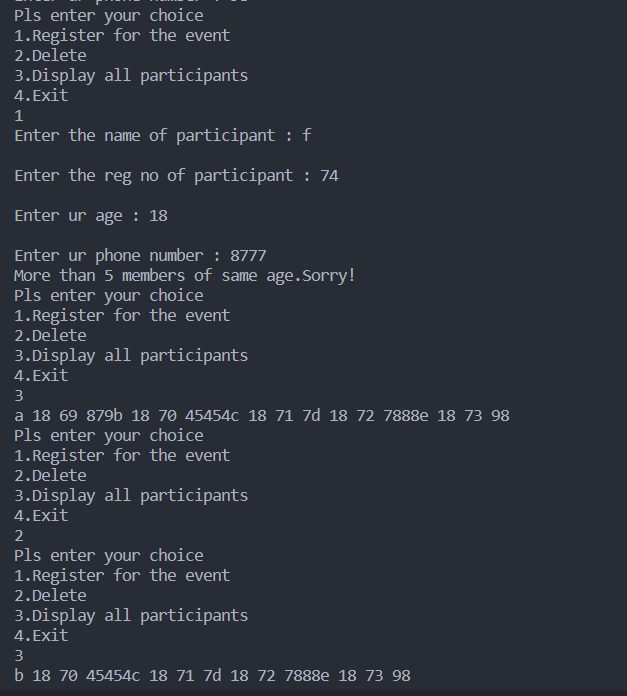
    return 1;

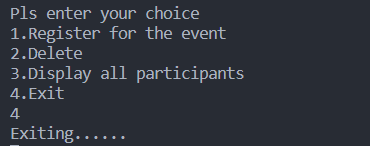
}

***OUTPUT***









4. Write a Program to create Queue of Patients waiting to see the Physician in a clinic. Insert

Patient details one by one into the Patient Queue in its appropriate position based on the

Age, irrespective of their arrival time. Patients should be allowed in the order from the

oldest to the youngest.

***CODE***

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <iostream>

using namespace std;

struct node

{

    int age;

    char name[20];

    char disease[30];

    char doctor\_assigned[30];

    struct node \*next;

} \*head = NULL;

int choice;

void enqueue();

void dequeue();

void display();

void enqueue()

{

    struct node \*temp;

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

    cout<<"Enter patient's name: ";

    cin>>temp->name;

    cout<<"Enter patient's age: ";

    cin>>temp->age;

    cout<<"Enter disease of patient: ";

    cin>>temp->disease;

    cout<<"Enter doctor assigned to patient: ";

    cin>>temp->doctor\_assigned;

    if (head == NULL)

    {

        head = temp;

        head->next = NULL;

    }

    else if (head->next == NULL)

    {

        if (head->age > temp->age)

        {

            head->next = temp;

            temp->next = NULL;

        }

        else

        {

            temp->next = head;

            head = temp;

        }

    }

    else

    {

        struct node \*r;

        r = head;

        while (r->next != NULL)

        {

            if (r->next->age > temp->age)

            {

                r = r->next;

            }

            else

            {

                break;

            }

        }

        if (r == head)

        {

            temp->next = head;

            head = temp;

        }

        else

        {

            temp->next = r->next;

            r->next = temp;

        }

    }

}

void dequeue ()

{

    struct node \*ptr;

    if(head == NULL)

    {

        printf("\nUNDERFLOW\n");

        return;

    }

    else

    {

        ptr = head;

        head = head -> next;

        free(ptr);

    }

}

void display()

{

    int c = 0;

    struct node \*r;

    r = head;

    while (r != NULL)

    {

        c++;

        printf("%d.%s(%d) , Disease: %s , Doctor Assigned: %s  \n", c, r->name, r->age,r->disease,r->doctor\_assigned);

        r = r->next;

    }

}

int main()

{

    while (choice != 4)

    {

        printf("\n1.Insert a patient\n");

        printf("2.Discharge Patient \n");

        printf("3.Display\n");

        printf("4.Exit\n");

        printf("Enter choice: ");

        scanf("%d", &choice);

        switch (choice)

        {

        case 1:

            enqueue();

            break;

        case 2:

            dequeue();

            break;

        case 3:

            display();

            break;

        case 4:

            printf("Exiting..\n");

            break;

        default:

            printf("Invalid input\n");

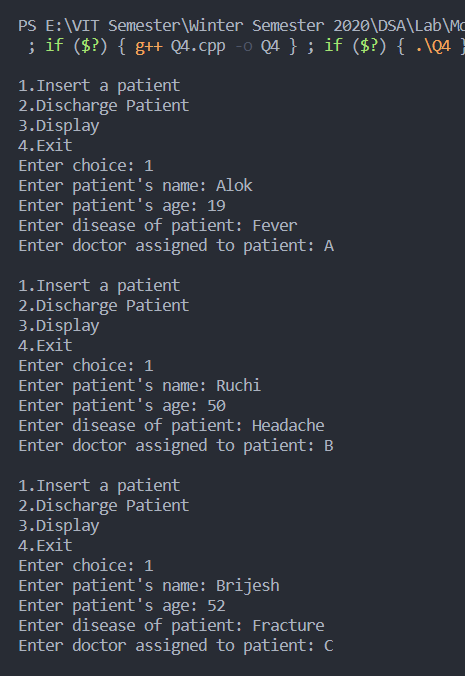
        };

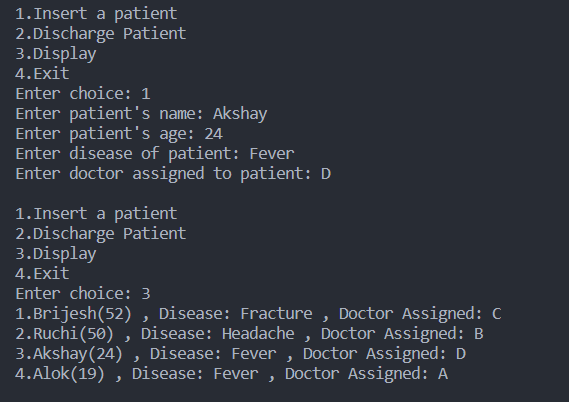
    }

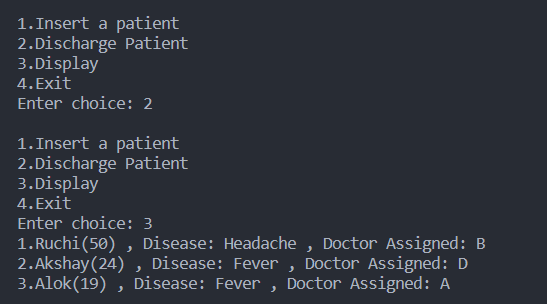
    return 1;

}

***OUTPUT***







5. Write the recursive code for the following

i)Linear Search

***CODE***

#include <stdio.h>

#include <iostream>

#include <string.h>

using namespace std;

int linear\_search(int *s*,int *arr*[],int *index*)

{

    if(*s*<=-1)

    {

        return -1;

    }

    else if(*arr*[*index*]==*s*)

    {

        return 1;

    }

    else

    {

        return linear\_search(*s*,*arr*,*index*+1);

    }

}

int main()

{

    int i,n;

    cout<<"The number of elements in arrray"<<endl;

    cin>>n;

    int arr[10];

    for(i=0;i<n;i++)

    {

        cin>>arr[i];

    }

    int search;

    cout<<"Enter the number to be searched"<<endl;

    cin>>search;

    int a=linear\_search(search,arr,0);

    if(a==1)

    {

        printf("Element is found");

    }

    else

    {

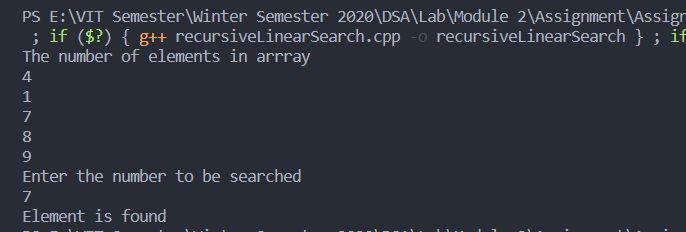
        printf("Element is not found");

    }

    return 1;

}

***OUTPUT***



ii) Binary Search

***CODE***

#include <stdio.h>

#include <iostream>

#include <string.h>

using namespace std;

void BinarySearch(int *arr*[],int *num*,int *first*,int *last*){

   int mid;

   if(*first* > *last*){

        printf("Number is not found");

   } else {

*/\* Calculate mid element \*/*

      mid = (*first* + *last*)/2;

*/\* If mid is equal to number we are searching \*/*

      if(*arr*[mid]==*num*){

            printf("Element is found at index %d ",mid);

            exit(0);

        }else if(*arr*[mid] > *num*){

            BinarySearch(*arr*, *num*, *first*, mid-1);

        }else{

            BinarySearch(*arr*, *num*, mid+1, *last*);

        }

    }

}

main(){

   int arr[100],beg,mid,end,i,n,num;

   printf("Enter the size of an array ");

   scanf("%d",&n);

   printf("Enter the values in sorted sequence \n");

   for(i=0;i<n;i++)

   {

       scanf("%d",&arr[i]);

   }

   beg=0;

   end=n-1;

   printf("Enter a value to be search: ");

   scanf("%d",&num);

   BinarySearch(arr,num,beg,end);

}

***OUTPUT***

