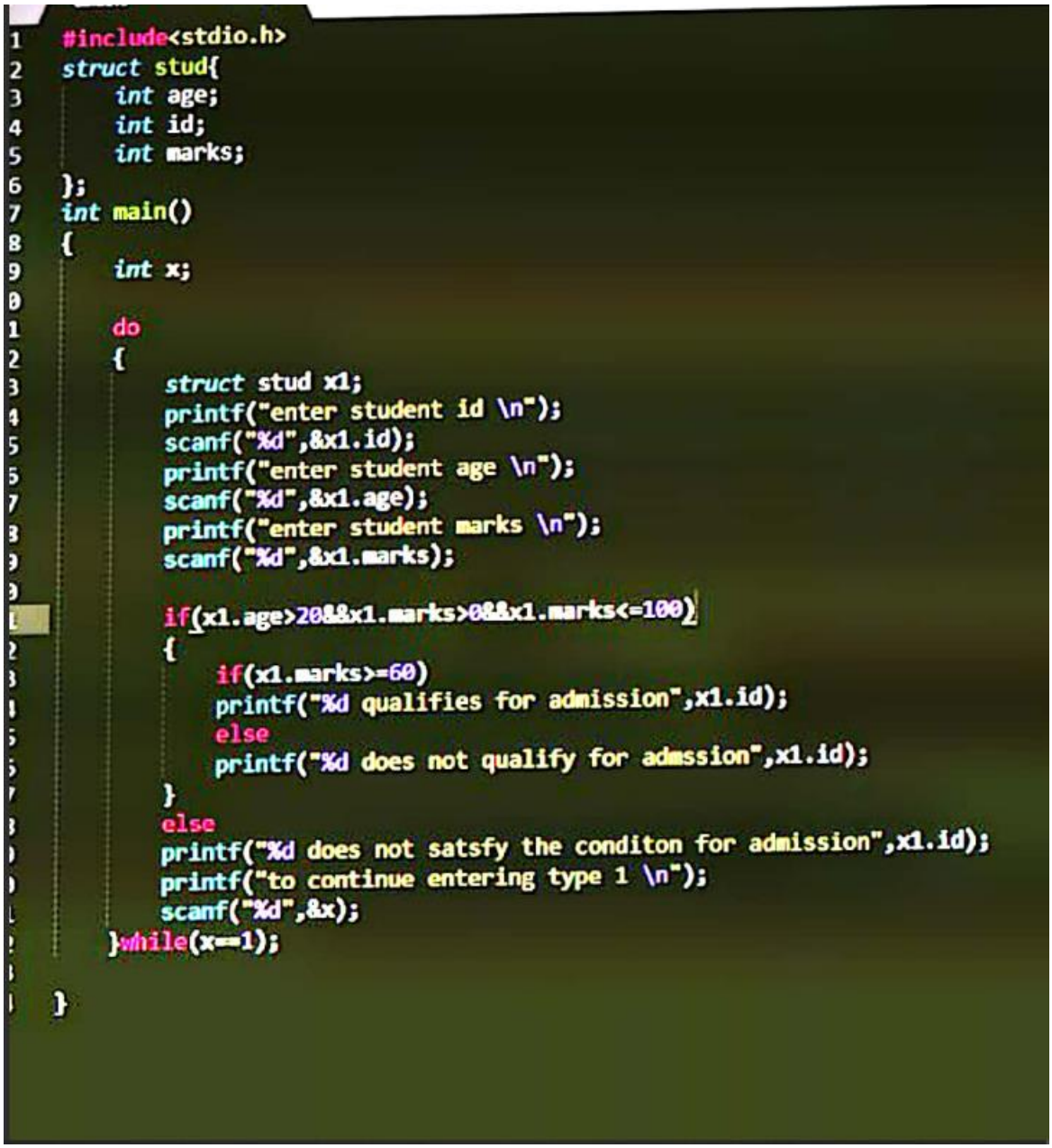
NAME : Rahul Suresh

USN : 1BM19CS204

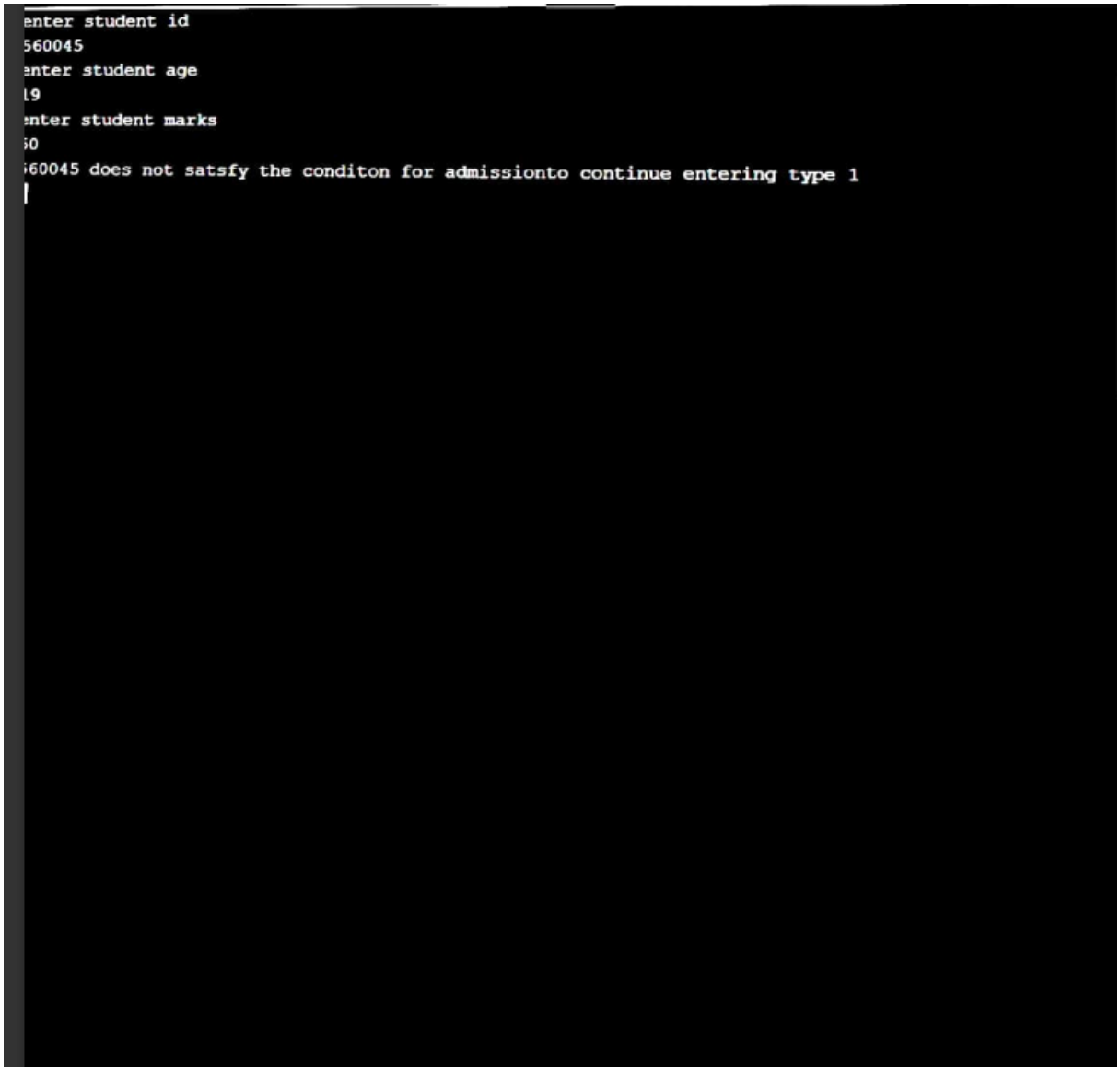
SUBJECT: DS LAB REPORT

ACADEMIC YEAR: 2019-2023

1. WAP for the below given scenario: A university wants to automate their admission process. Students are admitted based on the marks scored in a qualifying exam. A student is identified by student id, age and marks in qualifying exam. Data are valid, if: ● Age is greater than 20 ● Marks is between 0 and 100 (both inclusive) A student qualifies for admission, if ● Age and marks are valid and ● Marks is 65 or more



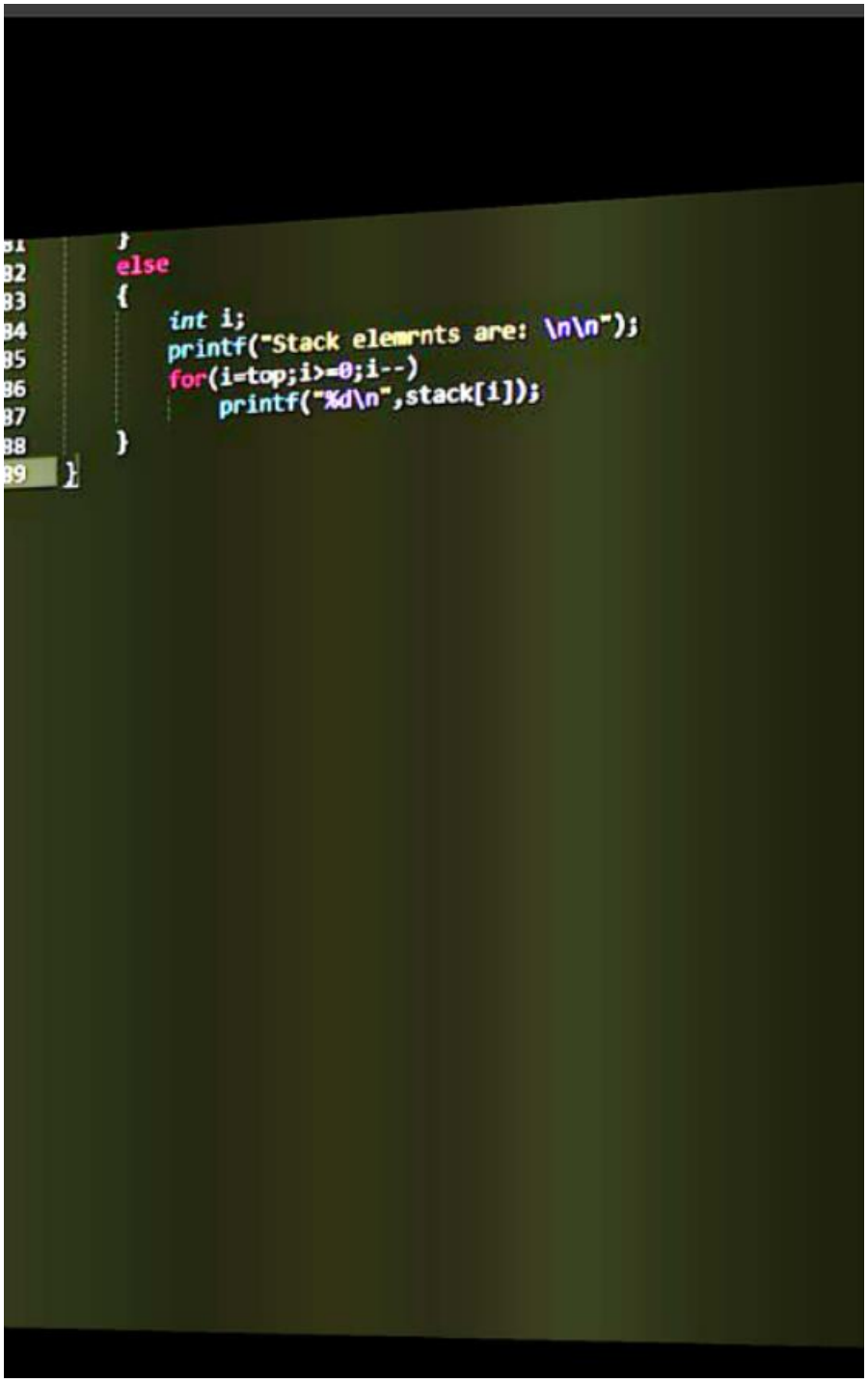
```c
#include<stdio.h>
struct stud{
    int age;
    int id;
    int marks;
};
int main()
{
    int x;

    do
    {
        struct stud x1;
        printf("enter student id \n");
        scanf("%d",&x1.id);
        printf("enter student age \n");
        scanf("%d",&x1.age);
        printf("enter student marks \n");
        scanf("%d",&x1.marks);

        if(x1.age>20&&x1.marks>0&&x1.marks<=100)
        {
            if(x1.marks>=60)
            printf("%d qualifies for admission",x1.id);
            else
            printf("%d does not qualify for admssion",x1.id);
        }
        else
        printf("%d does not satsfy the conditon for admission",x1.id);
        printf("to continue entering type 1 \n");
        scanf("%d",&x);
    }while(x==1);

}
```



2. Write a program to simulate the working of stack using an array with the following : a) Push b) Pop c) Display The program should print appropriate messages for stack overflow, stack underflow

```c
#include<stdio.h>
#include<stdlib.h>
#define SIZE 10

void push(int);
void pop();
void display();

int stack[SIZE],top=-1;

int main()
{
    int n,choice;
    while(1)
    {
        printf("\nMENU\n\n");
        printf("(1)Push\n");
        printf("(2)Pop\n");
        printf("(3)Display\n");
        printf("(4)Exit\n");
        printf("Enter your choice: \n\n");
        scanf("%d",&choice);

        switch(choice)
        {
            case 1:
                printf("Enter the value to be inserted:");
                scanf("%d",&n);
                push(n);
                break;

            case 2:
                pop();
                break;

            case 3:
                display();
                break;

            case 4:
```

```
                exit(0);

            default: printf("Incorrect Selection.Select Again!\n\n");
        }
    }
    return 0;
}

void push(int n)
{
    if(top==SIZE-1)
    {
        printf("Stack is Full.Insertion is not possible!\n\n");
    }
    else
    {
        top++;
        stack[top]=n;
        printf("Insertion Successful\n\n");
    }
}

void pop()
{
    if(top==-1)
    {
        printf("Stack is empty.Deletion is not possible!\n\n");
    }
    else
    {
        printf("Deleted: %d\n\n",stack[top]);
        top--;
    }
}

void display()
{
    if(top==-1)
    {
        printf("Stack is Empty\n\n");
```


}
else
{
int i;
printf("Stack elemrnts are: \n\n");
for(i=top;i>=0;i--)
printf("%d\n",stack[i]);
}
}

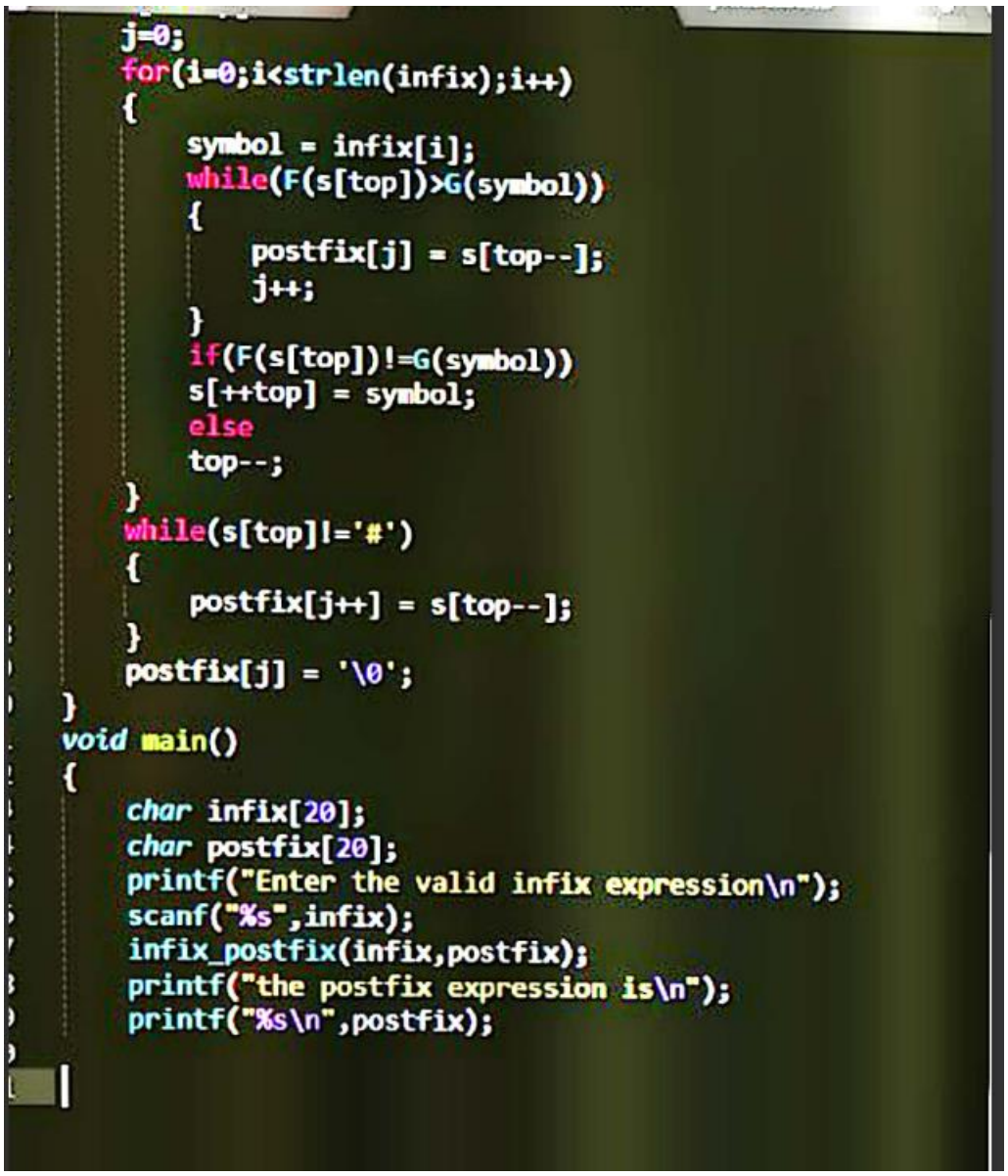
```
Enter your choice
1. Push
2. Pop
3. Display
1
Enter the element to be pushed
3
Do you want to continue:click-1
1
Enter your choice
1. Push
2. Pop
3. Display
1
Enter the element to be pushed
5
Do you want to continue:click-1
1
Enter your choice
1. Push
2. Pop
3. Display
1
Enter the element to be pushed
7
Do you want to continue:click-1
1
Enter your choice
1. Push
2. Pop
3. Display
3
The stack elements
```

input

```
2. Pop
3. Display
1
Enter the element to be pushed
7
Do you want to continue:click-1
1
Enter your choice
1. Push
2. Pop
3. Display
3
The stack elements
7       5       3
Do you want to continue:click-1
1
Enter your choice
1. Push
2. Pop
3. Display
2
Poped element is 7
Do you want to continue:click-1
1
Enter your choice
1. Push
2. Pop
3. Display
3
The stack elements
5       3
Do you want to continue:click-1
```

3. WAP to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and the binary operators + (plus), - (minus), * (multiply) and / (divide)

```c
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
int F(char symbol)
{
    switch(symbol)
    {
        case '+':
        case '-': return 2;
        case '*':
        case '/': return 4;
        case '^':
        case '$':return 5;
        case '(': return 0;
        case '#': return -1;
        default: return 8;
    }
}
int G(char symbol)
{
    switch(symbol)
    {
        case '+':
        case '-': return 1;
        case '*':
        case '/': return 3;
        case '^':
        case '$':return 6;
        case '(': return 9;
        case ')': return 0;
        default: return 7;
    }

}
void infix_postfix(char infix[],char postfix[])
{
    int top,i,j;
    char s[30],symbol;
    top=-1;
    s[++top] = '#';
```

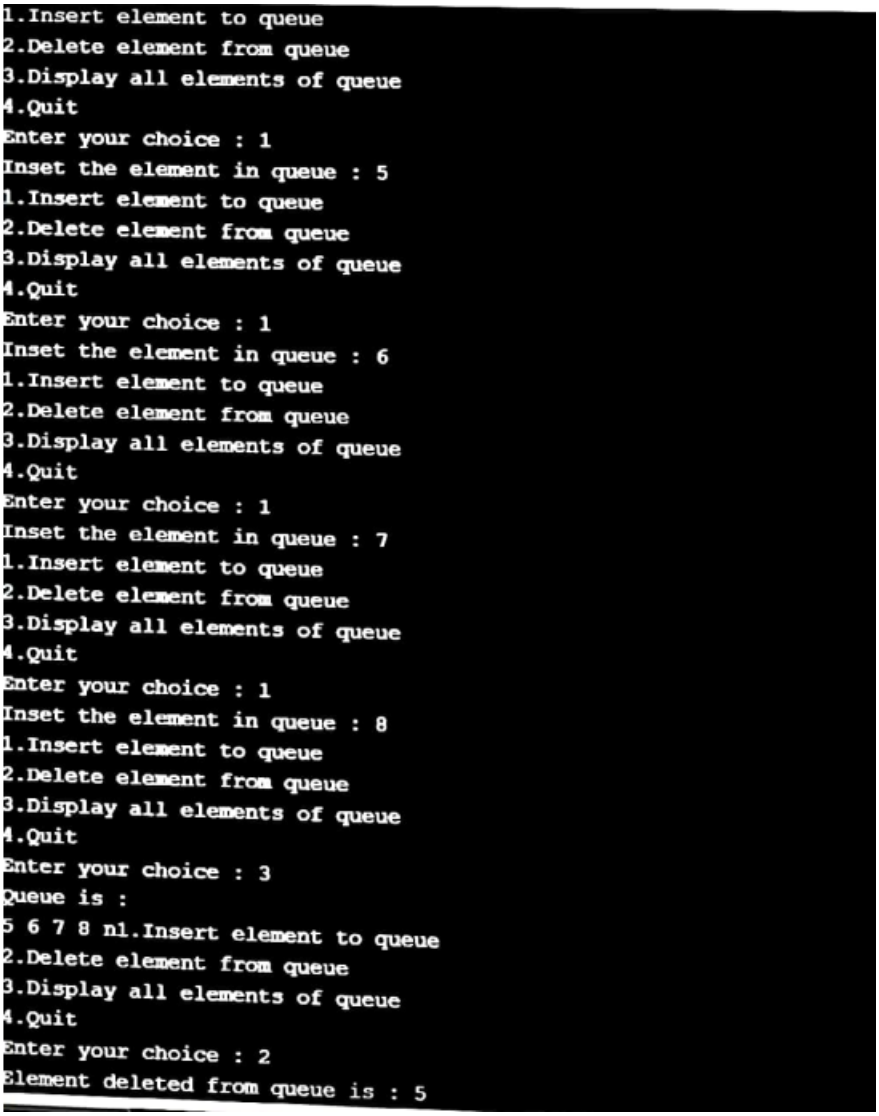


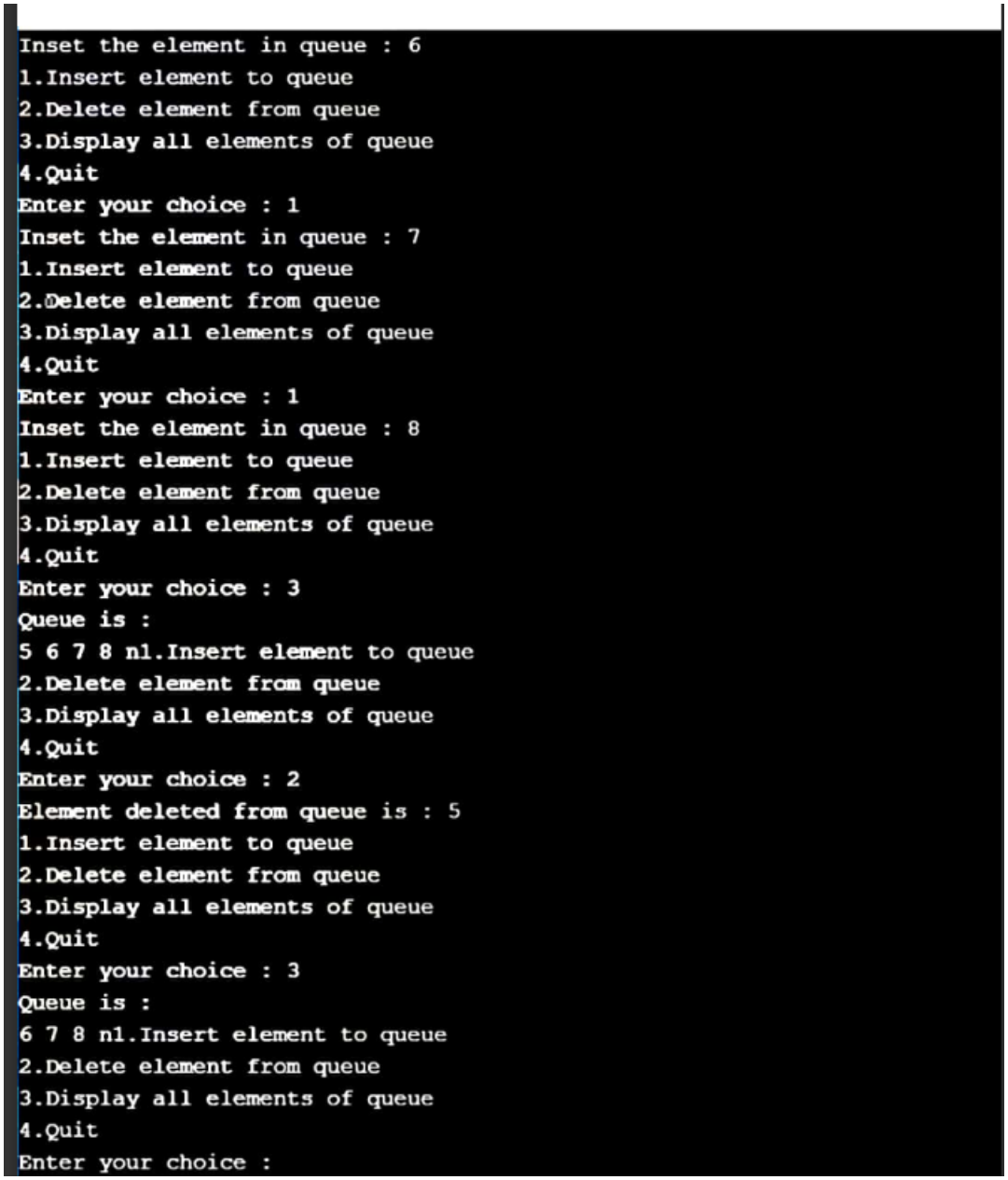
```
    j=0;
    for(i=0;i<strlen(infix);i++)
    {
        symbol = infix[i];
        while(F(s[top])>G(symbol))
        {
            postfix[j] = s[top--];
            j++;
        }
        if(F(s[top])!=G(symbol))
        s[++top] = symbol;
        else
        top--;
    }
    while(s[top]!='#')
    {
        postfix[j++] = s[top--];
    }
    postfix[j] = '\0';
}
void main()
{
    char infix[20];
    char postfix[20];
    printf("Enter the valid infix expression\n");
    scanf("%s",infix);
    infix_postfix(infix,postfix);
    printf("the postfix expression is\n");
    printf("%s\n",postfix);
```

4. WAP to simulate the working of a queue of integers using an array. Provide the following operations a) Insert b) Delete c) Display The program should print appropriate messages for queue empty and queue overflow conditions

```c
#include <stdio.h>
#include<stdlib.h>
#define MAX 50
void insert();
void delete();
void display();
int queue_array[MAX];
int rear = - 1;
int front = - 1;
int main()
{
int choice;
while (1)
{
printf("1.Insert element to queue \n");
printf("2.Delete element from queue \n");
printf("3.Display all elements of queue \n");
printf("4.Quit \n");
printf("Enter your choice : ");
scanf("%d", &choice);
switch(choice)
{
case 1:
insert();
break;
case 2:
delete();
break;
case 3:
display();
break;
case 4:
exit(1);
default:
printf("Wrong choice \n");
}
}
}
void insert()
{
```

```
{
int item;
if(rear == MAX - 1)
printf("Queue Overflow \n");
else
{
if(front== - 1)
front = 0;
printf("Inset the element in queue : ");
scanf("%d", &item);
rear = rear + 1;
queue_array[rear] = item;
}
}
void delete()
{
if(front == - 1 || front > rear)
{
printf("Queue Underflow \n");
return;
}
else
{
printf("Element deleted from queue is : %d\n", queue_array[front]);
front = front + 1;
}
}
void display()
{
int i;
if(front == - 1)
printf("Queue is empty \n");
else
{
printf("Queue is : \n");
for(i = front; i <= rear; i++)
printf("%d ", queue_array[i]);
printf("n");
}
}
```


1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 1
Inset the element in queue : 5
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 1
Inset the element in queue : 6
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 1
Inset the element in queue : 7
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 1
Inset the element in queue : 8
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 3
Queue is :
5 6 7 8 n1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 2
Element deleted from queue is : 5



5. WAP to simulate the working of a circular queue of integers using an array. Provide the following operations a) Insert b) Delete c) Display The program should print appropriate messages for queue empty and queue overflow conditions

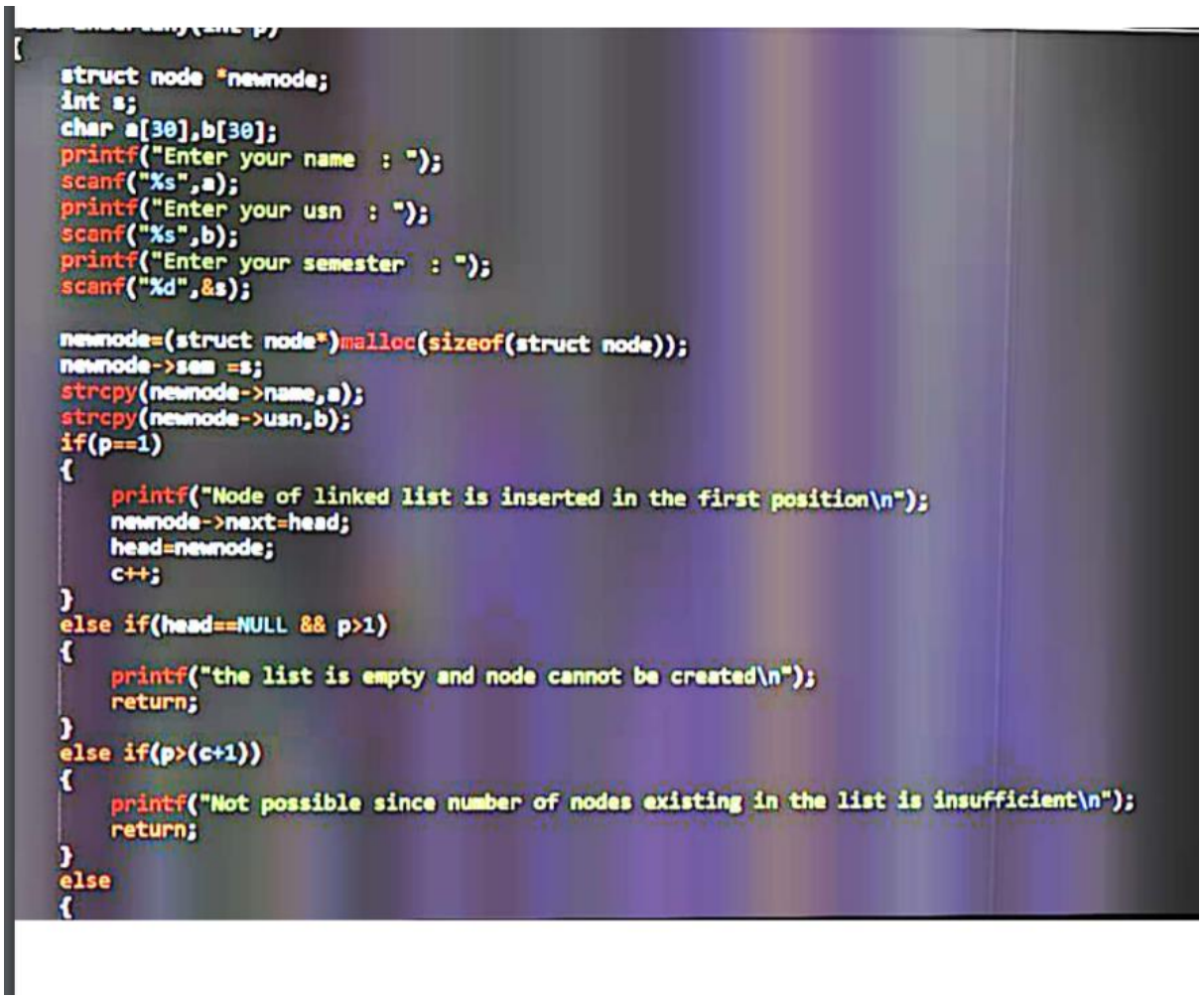
```c
#include<stdio.h>
# define max 3
void enqueue( int q[], int *f, int *r)
{
    if(*r-*f==max-1|| *r==*f-1)
    printf(" Queue is full\n\n");
    else
        {
        if(*r==-1)
            *f=*r=0;
        else
            *r=(*r+1)%max;
        printf("Enter the element:\n");
        scanf("%d", (&q[*r]));

        }
}


void dequeue(int q[], int *f, int *r)
{
    if (*f ==-1)
        printf(" Queue is empty\n\n");
    else
        {
        printf("%d is deleted\n", q[*f]);
        if (*f==*r)
            *f=*r=-1;
        else
            *f=(*f+1)%max;
        }
}

void display (int q[], int *f, int *r)
{
    if(*f==-1)
        printf("Queue is empty\n\n");
    else
        {
            for(int i=*f;;i++)
            {
                i=i%max;
                printf("%d ", q[i]);
                if (*r==i)
                    break;
            }
        printf("\n");
        }
```

```
}


int main()
{
    int choice, f=-1, r=-1, q[max];
    do
    {

        printf(" 1: Insert \n 2:Delete \n 3: Display\n 4: Exit\n");
        printf("Enter your choice\n");
        scanf("%d", &choice);
        switch(choice)
        {
            case 1: enqueue(q, &f, &r);
                break;
            case 2: dequeue(q, &f, &r);
                break;
            case 3: display( q, &f, &r);
                break;
            case 4:
                break;
            default: printf("INVALID CHOICE\n");
        }
    }while(choice!=4);
}
```

```
 1: Insert
 2:Delete
 3: Display
 4: Exit
Enter your choice
1
Enter the element:
4
 1: Insert
 2:Delete
 3: Display
 4: Exit
Enter your choice
1
Enter the element:
6
 1: Insert
 2:Delete
 3: Display
 4: Exit
Enter your choice
1
Enter the element:
8
 1: Insert
 2:Delete
 3: Display
 4: Exit
Enter your choice
3
4 6 8
 1: Insert
 2:Delete
 3: Display
 4: Exit
Enter your choice
```

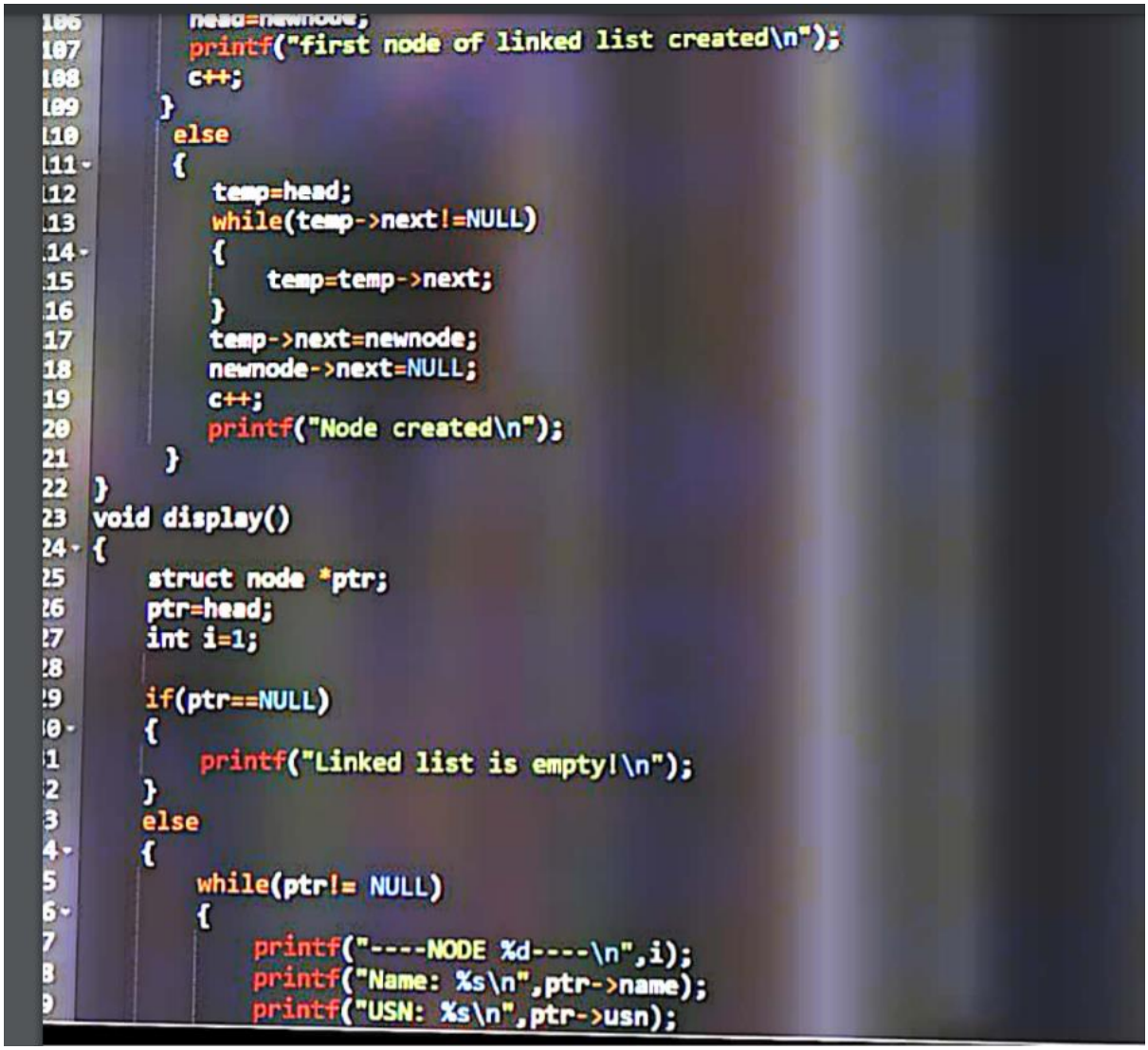
6. WAP to Implement Singly Linked List with following operations a) Create a linked list. b) Insertion of a node at first position, at any position and at end of list. c) Deletion of first element, specified element and last element in the list. d) Display the contents of the linked list.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct node
{
    int sem;
    char name[50];
    char usn[50];
    struct node *next;
};
struct node *head= NULL;
int c=0;
void Insertbegining()
{
    struct node *newnode;
    int s;
    char a[50],b[50];
    printf("Enter your name  : ");
    scanf("%s",a);
    printf("Enter your usn  : ");
    scanf("%s",b);
    printf("Enter your semester  : ");
    scanf("%d",&s);

    newnode=(struct node*)malloc(sizeof(struct node));
    newnode->sem =s;
    strcpy(newnode->name,a);
    strcpy(newnode->usn,b);

    newnode->next=head;
    head=newnode;
    c++;
    printf("Node created\n");
}
void Insertany(int p)
{
    struct node *newnode;
```



```
{
    struct node *newnode;
    int s;
    char a[30],b[30];
    printf("Enter your name  : ");
    scanf("%s",a);
    printf("Enter your usn  : ");
    scanf("%s",b);
    printf("Enter your semester  : ");
    scanf("%d",&s);

    newnode=(struct node*)malloc(sizeof(struct node));
    newnode->sem =s;
    strcpy(newnode->name,a);
    strcpy(newnode->usn,b);
    if(p==1)
    {
        printf("Node of linked list is inserted in the first position\n");
        newnode->next=head;
        head=newnode;
        c++;
    }
    else if(head==NULL && p>1)
    {
        printf("the list is empty and node cannot be created\n");
        return;
    }
    else if(p>(c+1))
    {
        printf("Not possible since number of nodes existing in the list is insufficient\n");
        return;
    }
    else
    {
```

```
70          struct node *temp2;
71          struct node *temp2;
72          int count=1;
73          temp1=head;
74          while(count<(p-1))
75          {
76              temp1= temp1->next;
77              count++;
78          }
79          temp2= temp1->next;
80          temp1->next=newnode;
81          newnode->next=temp2;
82          c++;
83          printf("Node inserted at %d position in linked list\n",p);
84      }
85  }
86
87  void Insertend()
88  {
89      struct node *newnode;
90      struct node *temp;
91      int s;
92      char n[30],u[30];
93      printf("Enter your name  : ");
94      scanf("%s",n);
95      printf("Enter your semester  : ");
96      scanf("%d",&s);
97      printf("Enter your usn  : ");
98      scanf("%s",u);
99      newnode=(struct node*)malloc(sizeof(struct node));
00      newnode->sem =s;
01      strcpy(newnode->name,n);
02      strcpy(newnode->usn,u);
03      if (head==NULL)
04      {
```

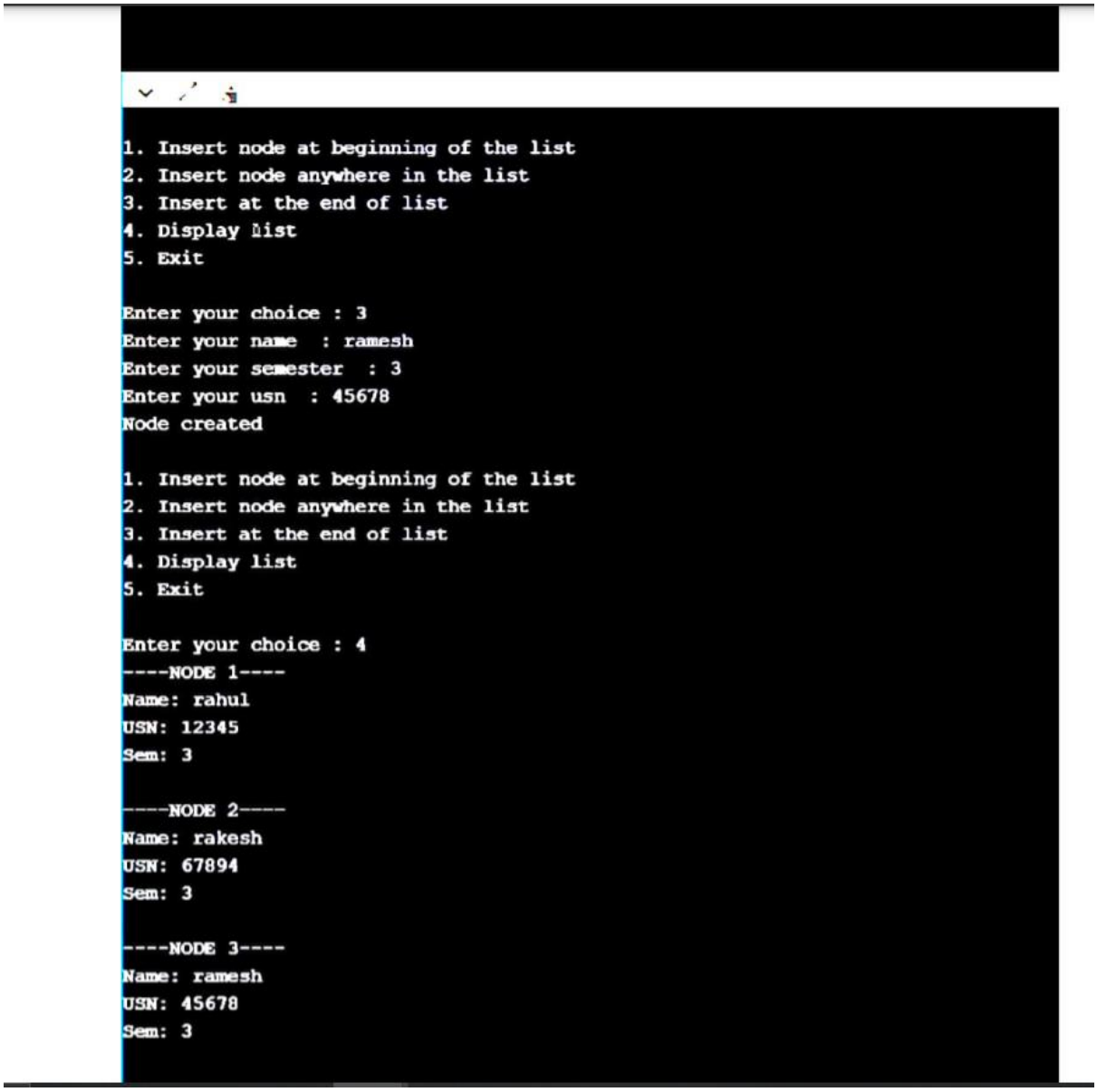


```
        head=newnode;
        printf("first node of linked list created\n");
        c++;
      }
       else
       {
          temp=head;
          while(temp->next!=NULL)
          {
              temp=temp->next;
          }
          temp->next=newnode;
          newnode->next=NULL;
          c++;
          printf("Node created\n");
       }
}
void display()
{
    struct node *ptr;
    ptr=head;
    int i=1;

    if(ptr==NULL)
    {
        printf("Linked list is empty!\n");
    }
    else
    {
        while(ptr!= NULL)
        {
            printf("----NODE %d----\n",i);
            printf("Name: %s\n",ptr->name);
            printf("USN: %s\n",ptr->usn);
```

```
            printf("\n");
            i++;
            ptr=ptr->next;
        }
    }
}

int main()
{
    int choice,pos;
    do
    {
        printf("\n1. Insert node at beginning of the list\n2. Insert node anywhere in the list\n3. Insert at the end of list\n4. Display list\n5. Exit\n");
        printf("\nEnter your choice : ");
        scanf("%d",&choice);
        if(choice==5)
        break;
        switch(choice)
        {
            case 1:
            Insertbegining();
            break;

            case 2:
            printf("Enter in which position of the list you want to enter your node\n");
            scanf("%d",&pos);
            Insertany(pos);
            break;

            case 3:
            Insertend();
```

```
    int choice,pos;
    do
    {

        printf("\n1. Insert node at beginning of the list\n2. Insert node anywhere in the list\n3. Insert at the end of list\n4. Display list\n5. Exit\n")
        printf("\nEnter your choice : ");
        scanf("%d",&choice);
        if(choice==5)
        break;
        switch(choice)
        {
            case 1:
            Insertbegining();
            break;

            case 2:
            printf("Enter in which position of the list you want to enter your node\n");
            scanf("%d",&pos);
            Insertany(pos);
            break;

            case 3:
            Insertend();
            break;

            case 4:
            display();
            break;

            default:
            printf("Wrong choice!\n");
            break;
        }
    }while(choice!=5);
    return 0;
}
```

```
1. Insert node at beginning of the list
2. Insert node anywhere in the list
3. Insert at the end of list
4. Display list
5. Exit

Enter your choice : 1
Enter your name  : rahul
Enter your usn  : 12345
Enter your semester  : 3
Node created

1. Insert node at beginning of the list
2. Insert node anywhere in the list
3. Insert at the end of list
4. Display list
5. Exit

Enter your choice : 2
Enter in which position of the list you want to enter your node
2
Enter your name  : rakesh
Enter your usn  : 67894
Enter your semester  : 3
Node inserted at 2 position in linked list

1. Insert node at beginning of the list
2. Insert node anywhere in the list
3. Insert at the end of list
4. Display list
5. Exit

Enter your choice : 3
Enter your name  : ramesh
Enter your semester  : 3
```



7. WAP Implement Single Link List with following operations a) Sort the linked list. b) Reverse the linked list. c) Concatenation of two linked lists d) implement Stack & Queues using Linked Representation

```c
#include <stdlib.h>
#include <string.h>
struct node
{
    int sem;
    struct node *next;
};
struct node *head= NULL;
struct node *head2= NULL;
int c=0;
void Insert()
{
    struct node *newnode;
    struct node *temp;
    int s;
    printf("Enter integer  : ");
    scanf("%d",&s);
    newnode=(struct node*)malloc(sizeof(struct node));
    newnode->sem =s;
    if (head==NULL)
    {
      newnode->next=NULL;
      head=newnode;
      printf("first node of linked list created\n");
      c++;
    }
     else
     {
        temp=head;
        while(temp->next!=NULL)
        {
            temp=temp->next;
        }
        temp->next=newnode;
        newnode->next=NULL;
        c++;
        printf("Node created\n");
     }
}
void Insert2()
{
    struct node *newnode;
    struct node *temp;
    int s,y;
    printf("enter elements to create list 2\n");
    do
    {
```

```
        scanf("%d",&s);
        newnode=(struct node*)malloc(sizeof(struct node));
        newnode->sem =s;
        if (head2==NULL)
        {
          newnode->next=NULL;
          head2=newnode;
          printf("first node of linked list created\n");
          c++;
        }
         else
         {
            temp=head2;
            while(temp->next!=NULL)
            {
                temp=temp->next;
            }
            temp->next=newnode;
            newnode->next=NULL;
            c++;
            printf("Node created\n");
         }
         printf("do u want to continue adding:0 or 1\n");
         scanf("%d",&y);
        }while(y!=0);
    }


    void bubbleSort()
    {
        int swapped, i;
        struct node *ptr1;
        struct node *lptr = NULL;


        if (head == NULL)
            return;

        do
        {
            swapped = 0;
            ptr1 = head;

            while (ptr1->next != lptr)
            {
                if (ptr1->sem > ptr1->next->sem)
                {
                    int temp = ptr1->sem;
```

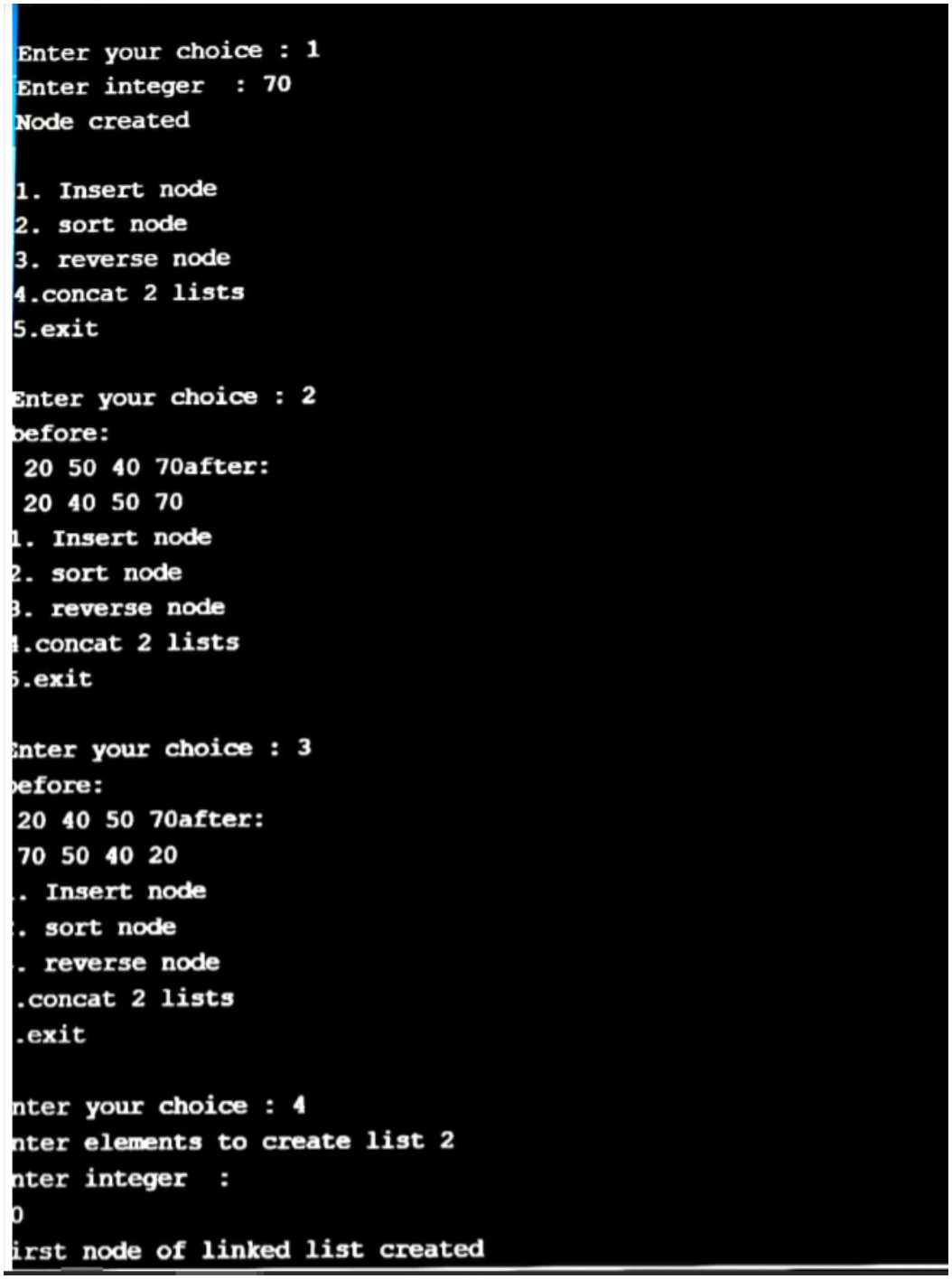
```
                swapped = 1;
            }
            ptr1 = ptr1->next;
        }
        lptr = ptr1;
    }
    while (swapped);
}

void reverse()
{
    struct node* prev = NULL;
    struct node* current = head;
    struct node* next = NULL;
    while (current != NULL) {
        next = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }
    head= prev;
}

void concat()
{
        struct node *ptr;
        if(head==NULL)
        {
                head=head2;
        }
        if(head2==NULL)
                {
                 head2=head;
                }
        ptr=head;
        while(ptr->next!=NULL)
                ptr=ptr->next;
        ptr->next=head2;
}
void display1()
{
    struct node *ptr;
    ptr=head;
    int i=1;

    if(ptr==NULL)
```

```
    {
        printf("Linked list is empty!\n");
    }
    else
    {
        while(ptr!= NULL)
        {
            printf(" %d",ptr->sem);
            i++;
            ptr=ptr->next;
        }

    }

}

int main()
{
    int choice,pos;
    do
    {

        printf("\n1. Insert node \n2. sort node\n3. reverse node\n4.concat 2 lists \n5.exit\n");
        printf("\nEnter your choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            Insert();
            break;

            case 2:
            printf("before:\n");
            display1();
            bubbleSort();
            printf("after:\n");
            display1();
            break;

            case 3:
            printf("before:\n");
            display1();
            reverse();
            printf("after:\n");

            display1();
            break;
```

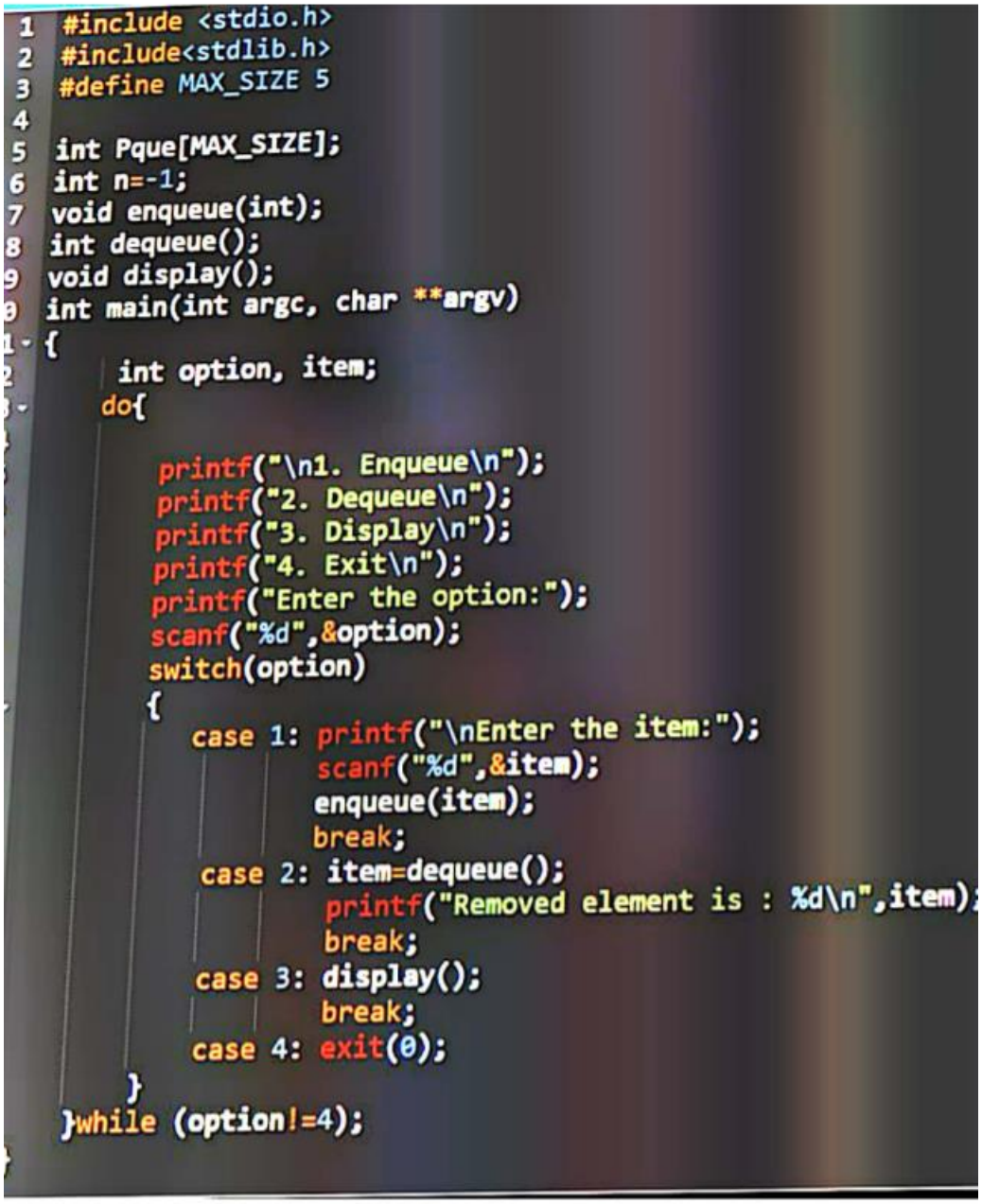
```
        printf("\nEnter your choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            Insert();
            break;

            case 2:
            printf("before:\n");
            display1();
            bubbleSort();
            printf("after:\n");
            display1();
            break;

            case 3:
            printf("before:\n");
            display1();
            reverse();
            printf("after:\n");

            display1();
            break;

            case 4:
            Insert2();
            concat();
            display1();
            break;

            case 5:
            break;

            default:
            printf("Wrong choice!\n");
            break;
        }
    }while(choice!=5);
    return 0;
}
```

```
5.exit

Enter your choice : 3
before:
 20 40 50 70after:
 70 50 40 20
1. Insert node
2. sort node
3. reverse node
4.concat 2 lists
5.exit

Enter your choice : 4
enter elements to create list 2
Enter integer  :
40
first node of linked list created
do u want to continue adding:0 or 1
1
Enter integer  :
60
Node created
do u want to continue adding:0 or 1
1
Enter integer  :
80
Node created
do u want to continue adding:0 or 1
0
 70 50 40 20 40 60 80
1. Insert node
2. sort node
3. reverse node
4.concat 2 lists
5.exit
```

```
1. Insert node
2. sort node
3. reverse node
4.concat 2 lists
5.exit

Enter your choice : 1
Enter integer  : 20
first node of linked list created

1. Insert node
2. sort node
3. reverse node
4.concat 2 lists
5.exit

Enter your choice : 1
Enter integer  : 50
Node created

1. Insert node
2. sort node
3. reverse node
4.concat 2 lists
5.exit

Enter your choice : 1
Enter integer  : 40
Node created

1. Insert node
2. sort node
3. reverse node
4.concat 2 lists
5.exit
```

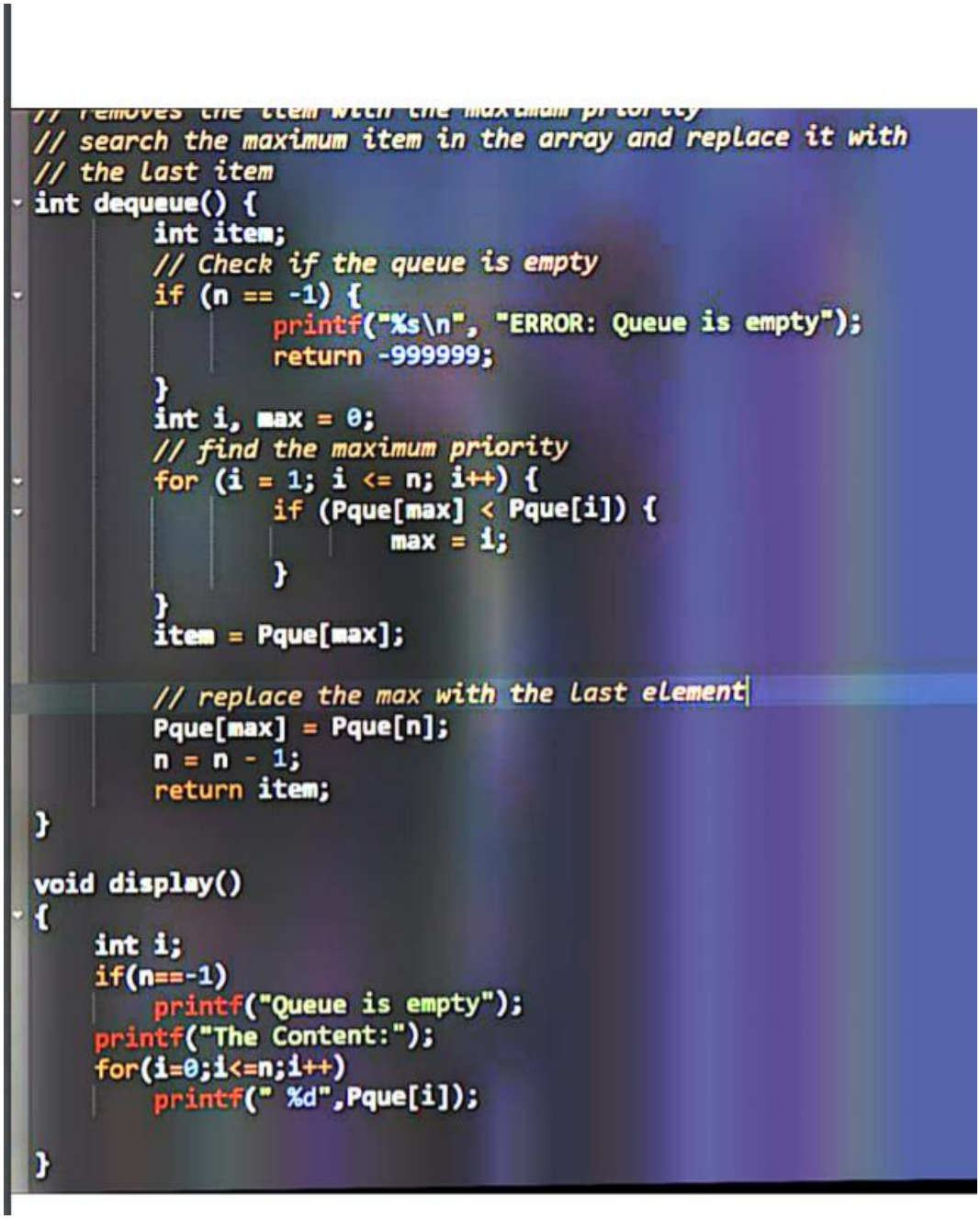


8.WAP to simulate the working of a priority queue of integers using an array. Provide the following operations a) Insert b) Delete c) Display The program should print appropriate messages for queue empty and queue overflow conditions



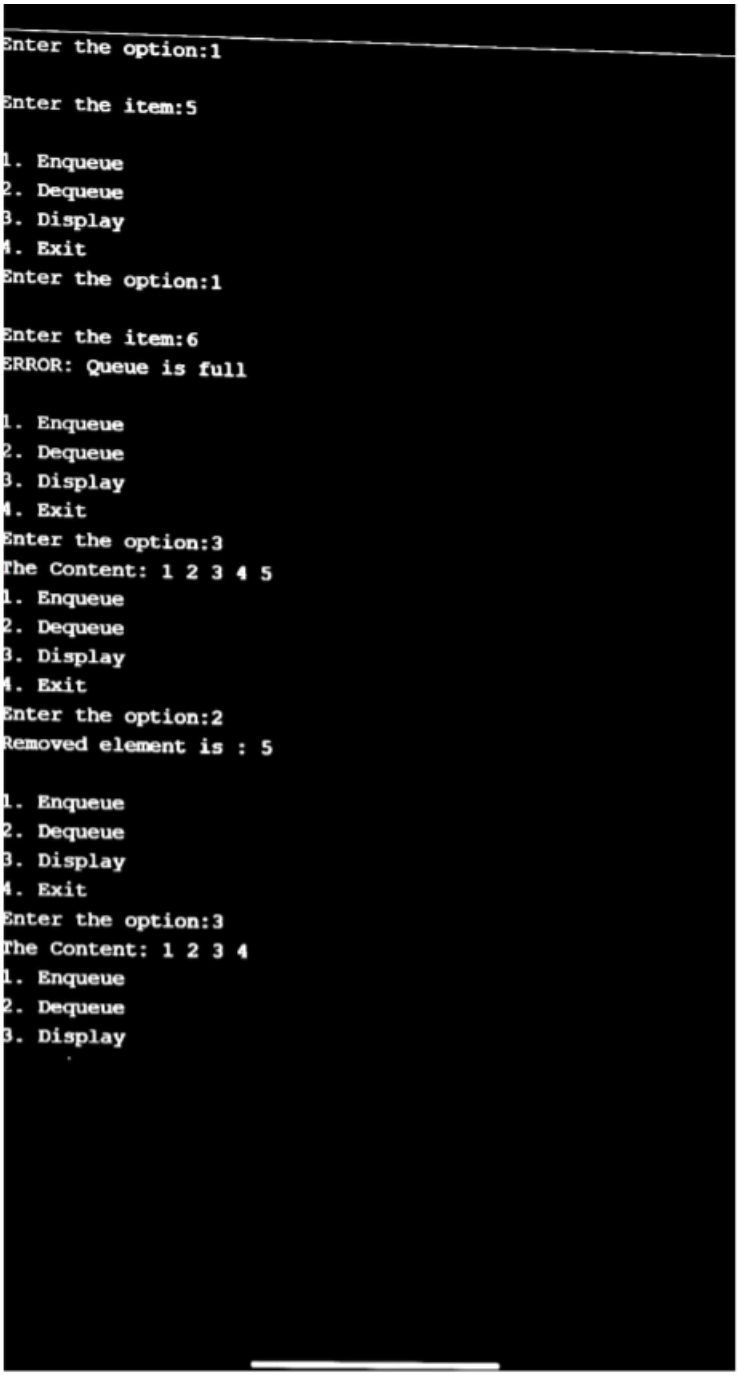
```c
#include <stdio.h>
#include<stdlib.h>
#define MAX_SIZE 5

int Pque[MAX_SIZE];
int n=-1;
void enqueue(int);
int dequeue();
void display();
int main(int argc, char **argv)
{
    int option, item;
    do{

        printf("\n1. Enqueue\n");
        printf("2. Dequeue\n");
        printf("3. Display\n");
        printf("4. Exit\n");
        printf("Enter the option:");
        scanf("%d",&option);
        switch(option)
        {
            case 1: printf("\nEnter the item:");
                    scanf("%d",&item);
                    enqueue(item);
                    break;
             case 2: item=dequeue();
                     printf("Removed element is : %d\n",item);
                     break;
            case 3: display();
                    break;
            case 4: exit(0);
        }
    }while (option!=4);
}
```

```
36
37  void enqueue(int item) {
38          // Check if the queue is full
39          if (n == MAX_SIZE - 1) {
40                  printf("%s\n", "ERROR: Queue is full");
41                  return;
42          }
43          n++;
44          Pque[n] = item;
45  }
46
47  // removes the item with the maximum priority
48  // search the maximum item in the array and replace it with
49  // the last item
50  int dequeue() {
51          int item;
52          // Check if the queue is empty
53          if (n == -1) {
54                  printf("%s\n", "ERROR: Queue is empty");
55                  return -999999;
56          }
57          int i, max = 0;
58          // find the maximum priority
59          for (i = 1; i <= n; i++) {
60                  if (Pque[max] < Pque[i]) {
61                          max = i;
62                  }
63          }
64          item = Pque[max];
65
66          // replace the max with the last element
67          Pque[max] = Pque[n];
68          n = n - 1;
69          return item;
70  }
71
```



```
// removes the item with the maximum priority
// search the maximum item in the array and replace it with
// the last item
int dequeue() {
        int item;
        // Check if the queue is empty
        if (n == -1) {
                printf("%s\n", "ERROR: Queue is empty");
                return -999999;
        }
        int i, max = 0;
        // find the maximum priority
        for (i = 1; i <= n; i++) {
                if (Pque[max] < Pque[i]) {
                        max = i;
                }
        }
        item = Pque[max];

        // replace the max with the last element
        Pque[max] = Pque[n];
        n = n - 1;
        return item;
}

void display()
{
    int i;
    if(n==-1)
        printf("Queue is empty");
    printf("The Content:");
    for(i=0;i<=n;i++)
        printf(" %d",Pque[i]);

}
```

```
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter the option:1

Enter the item:1

1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter the option:1

Enter the item:2

1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter the option:1

Enter the item:3

1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter the option:1

Enter the item:4

1. Enqueue
2. Dequeue
3. Display
```

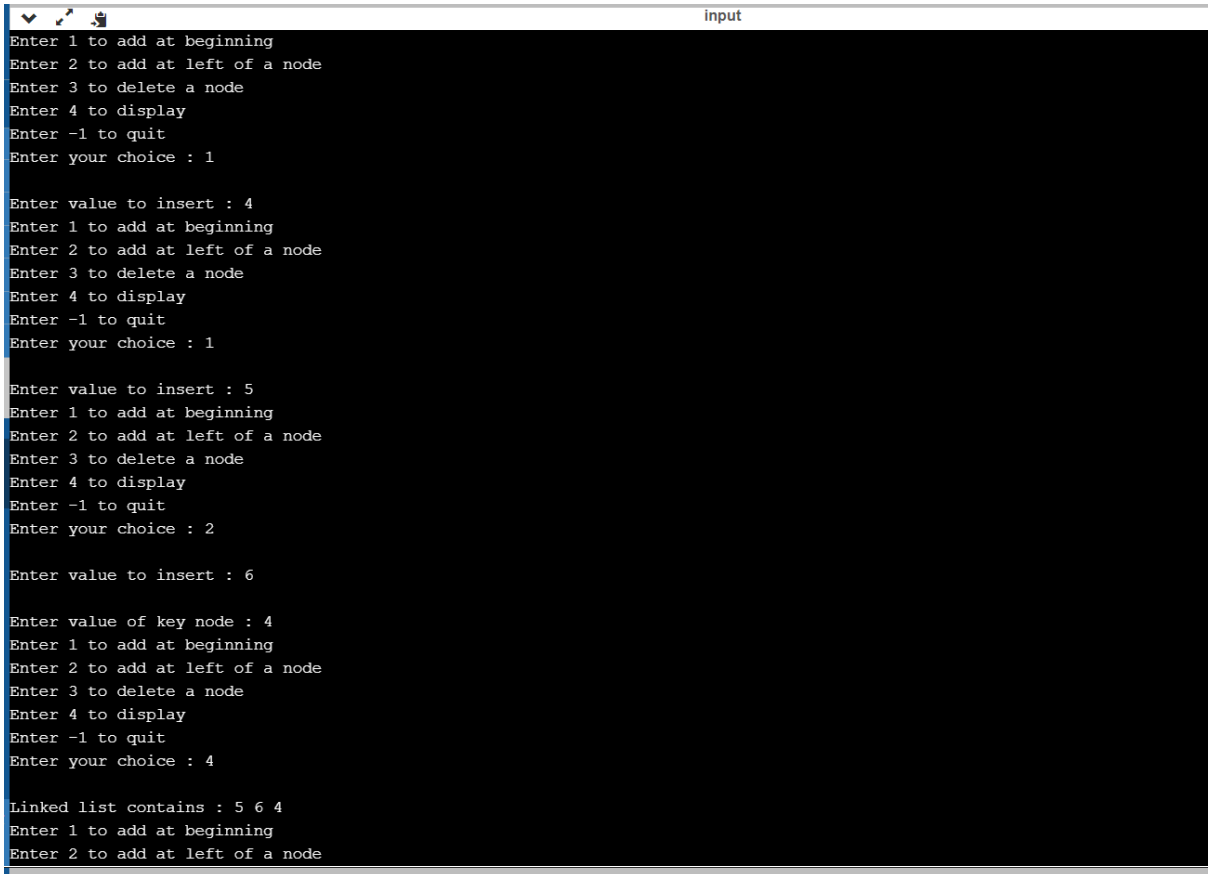


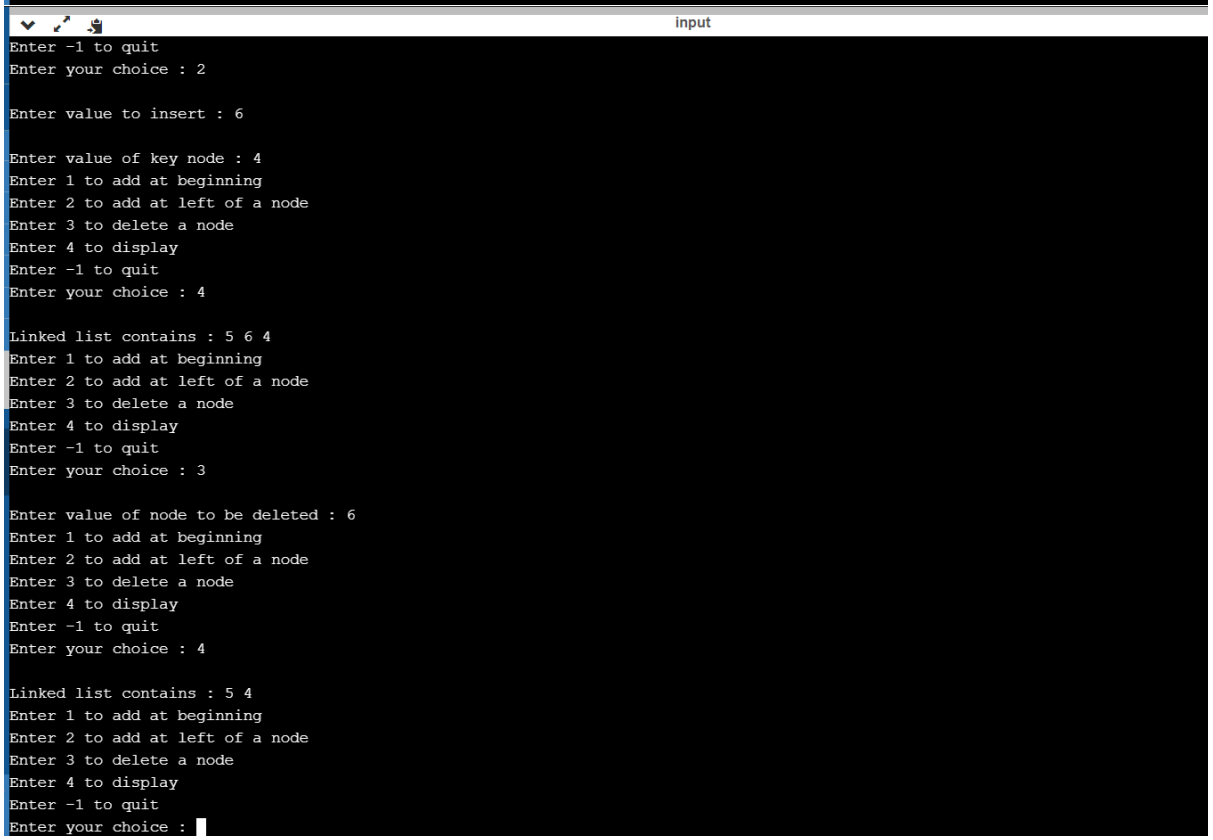
9. WAP Implement doubly link list with primitive operations a) Create a doubly linked list. b) Insert a new node to the left of the node. c) Delete the node based on a specific value d) Display the contents of the list

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

typedef struct Node {
    int value;
    struct Node* prev;
    struct Node* next;
}node;

node* head = NULL;

void add_beg(int value) {
    node* ptr = (node*) malloc(sizeof(node));
    ptr->value = value;
    ptr->prev = NULL;
    ptr->next = head;
    if(head!=NULL) {
        head->prev = ptr;
    }
    head = ptr;
}

void add_key(int value, int key) {
    node* tmp = head;
    while(tmp!=NULL) {
        if(tmp->value == key) {
            break;
        }
        tmp = tmp->next;
    }
    if(tmp==NULL) {
        printf("\nNo Match\n");
        return;
    }
    if(tmp==head) {
        add_beg(value);
        return;
    }
    node* ptr = (node*) malloc(sizeof(node));
    ptr->value = value;
    ptr->prev = tmp->prev;
    ptr->next = tmp;
    (tmp->prev)->next = ptr;
    tmp->prev = ptr;
}
void del_key(int key) {
    if(head == NULL) {
        printf("\nList is Empty\n");
        return;
    }
    node* tmp = head;
    while(tmp != NULL) {
        if(tmp -> value == key) {
            break;
        }
        tmp = tmp->next;
    }
    if(tmp==head) {
        if(head->next==NULL)
        {
            free(head);
            head=NULL;
            return;
        }
        head = head->next;
        free(head->prev);
        head->prev = NULL;
        return;
    }
    if(tmp==NULL) {
        printf("\nNo Match\n");
        return;
    }
    if(tmp->next==NULL) {
        tmp->prev->next = NULL;
```

```
        free(tmp);
        return;
    }
    tmp->next->prev = tmp->prev;
    tmp->prev->next = tmp->next;
    free(tmp);
}
void display() {
    if(head == NULL) {
        printf("\nList is Empty\n");
        return;
    }
    node* tmp = head;
    printf("\nLinked list contains : ");
    while(tmp!= NULL) {
        printf("%d ", tmp->value);
        tmp = tmp->next;
    }
    printf("\n");
}
void main() {
    int choice, value, key;
    while(1) {
        printf("Enter 1 to add at beginning\n");
        printf("Enter 2 to add at left of a node\n");
        printf("Enter 3 to delete a node\n");
        printf("Enter 4 to display\n");
        printf("Enter -1 to quit\n");
        printf("Enter your choice : ");
        scanf("%d", &choice);
        if(choice==-1)
            break;
        switch(choice) {
            case 1:
                printf("\nEnter value to insert : ");
                scanf("%d", &value);
                add_beg(value);
```

```
    while(1) {
        printf("Enter 1 to add at beginning\n");
        printf("Enter 2 to add at left of a node\n");
        printf("Enter 3 to delete a node\n");
        printf("Enter 4 to display\n");
        printf("Enter -1 to quit\n");
        printf("Enter your choice : ");
        scanf("%d", &choice);
        if(choice==-1)
            break;
        switch(choice) {
            case 1:
                printf("\nEnter value to insert : ");
                scanf("%d", &value);
                add_beg(value);
                break;
            case 2:
                printf("\nEnter value to insert : ");
                scanf("%d", &value);
                printf("\nEnter value of key node : ");
                scanf("%d", &key);
                add_key(value, key);
                break;
            case 3:
                printf("\nEnter value of node to be deleted : ");
                scanf("%d", &key);
                del_key(key);
                break;
            case 4:
                display();
                break;
            default:
                printf("\n\nWrong Input\n\n");
        }
    }
    printf("\n\n-----DONE-----\n\n");
}
```





10. binary tree program

```c
#include<stdio.h>
#include<stdlib.h>
struct node
{
     int data;
     struct node* left;
     struct node* right;
}*root1;

struct node *create()

{

 struct node *temp;

printf("\n Enter data:");

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

scanf("%d",&temp->data);

temp->left=temp->right=NULL;

return temp;

}

void insert(struct node *root,struct node *temp)

{


if(temp->data<root->data)

{

if(root->left!=NULL)
```

```c

insert(root->left,temp);

else

root->left=temp;

}

if(temp->data>root->data)

{

if(root->right!=NULL)

insert(root->right,temp);

else

root->right=temp;
}
}


void Postorder(struct node* node)
{
     if (node == NULL)
        return;


     Postorder(node->left);

     Postorder(node->right);


     printf("%d ", node->data);
}
```

```c

void Inorder(struct node* node)
{
    if (node == NULL)
        return;


    Inorder(node->left);

    printf("%d ", node->data);


    Inorder(node->right);
}


void Preorder(struct node* node)
{
    if (node == NULL)
        return;


    printf("%d ", node->data);


    Preorder(node->left);


    Preorder(node->right);
}
int main()
{
     int ch;
     struct node *temp;
     do
     {
     struct node *temp;
     do
     {
     printf("1.create\n2.insert\n3.preorder\n4.postorder\n5.inorder\n6.Exit\n");
     scanf("%d",&ch);
     switch(ch)
     {
     case 1:
     root1=create();
     break;
     case 2:
     printf("enter the elem to be entered\n");
     temp=(struct node*)malloc(sizeof(struct node));
     scanf("%d",&temp->data);
     insert(root1,temp);
     break;
     case 3:
     Preorder(root1);
     printf("\n");
     break;
     case 4:
     Postorder(root1);
     printf("\n");
     break;
     case 5:
     Inorder(root1);
     printf("\n");
     break;
     case 6:
     break;
     default:
     printf("wrong entry");
  }
     }while(ch!=6);


```

```
1.create
2.insert
3.preorder
4.postorder
5.inorder
6.Exit
1

 Enter data:4
1.create
2.insert
3.preorder
4.postorder
5.inorder
6.Exit
2
enter the elem to be entered
5
1.create
2.insert
3.preorder
4.postorder
5.inorder
6.Exit
2
enter the elem to be entered
6
1.create
2.insert
3.preorder
4.postorder
5.inorder
6.Exit
2
enter the elem to be entered
7
1.create
2.insert
3.preorder
4.postorder
5.inorder
6.Exit
2
enter the elem to be entered
8
1.create
2.insert
3.preorder
4.postorder
5.inorder
6.Exit
2
enter the elem to be entered
9
1.create
2.insert
3.preorder
4.postorder
5.inorder
6.Exit
3
4 5 6 7 8 9
1.create
2.insert
3.preorder
4.postorder
5.inorder
6.Exit
```

```
2
enter the elem to be entered
9
1.create
2.insert
3.preorder
4.postorder
5.inorder
6.Exit
3
4 5 6 7 8 9
1.create
2.insert
3.preorder
4.postorder
5.inorder
6.Exit
4
9 8 7 6 5 4
1.create
2.insert
3.preorder
4.postorder
5.inorder
6.Exit
5
4 5 6 7 8 9
1.create
2.insert
3.preorder
4.postorder
5.inorder
6.Exit
```