

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

```
1  #include<stdio.h>
2  struct stud{
3      int age;
4      int id;
5      int marks;
6  };
7  int main()
8  {
9      int x;
10
11      do
12      {
13          struct stud x1;
14          printf("enter student id \n");
15          scanf("%d",&x1.id);
16          printf("enter student age \n");
17          scanf("%d",&x1.age);
18          printf("enter student marks \n");
19          scanf("%d",&x1.marks);
20
21          if(x1.age>20&&x1.marks>0&&x1.marks<=100)
22          {
23              if(x1.marks>=60)
24                  printf("%d qualifies for admission",x1.id);
25              else
26                  printf("%d does not qualify for admssion",x1.id);
27          }
28          else
29              printf("%d does not satsfy the conditon for admission",x1.id);
30          printf("to continue entering type 1 \n");
31          scanf("%d",&x);
32      }while(x==1);
33  }
```

```
enter student id
560045
enter student age
19
enter student marks
60
560045 does not satisfy the condition for admission to continue entering type 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

```
1  #include<stdio.h>
2  #include<stdlib.h>
3  #define SIZE 10
4
5  void push(int);
6  void pop();
7  void display();
8
9  int stack[SIZE],top=-1;
10
11 int main()
12 {
13     int n,choice;
14     while(1)
15     {
16         printf("\nMENU\n\n");
17         printf("(1)Push\n");
18         printf("(2)Pop\n");
19         printf("(3)Display\n");
20         printf("(4)Exit\n");
21         printf("Enter your choice: \n\n");
22         scanf("%d",&choice);
23
24         switch(choice)
25         {
26             case 1:
27                 printf("Enter the value to be inserted:");
28                 scanf("%d",&n);
29                 push(n);
30                 break;
31
32             case 2:
33                 pop();
34                 break;
35
36             case 3:
37                 display();
38                 break;
39
40             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");
    }
}

```

```
81     }
82     else
83     {
84         int i;
85         printf("Stack elements are: \n\n");
86         for(i=top; i>=0; i--)
87             printf("%d\n", stack[i]);
88     }
89 }
```

```
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)

```
1  #include<stdio.h>
2  #include<string.h>
3  #include<stdlib.h>
4  int F(char symbol)
5  {
6      switch(symbol)
7      {
8          case '+':
9          case '-': return 2;
10         case '*':
11         case '/': return 4;
12         case '^':
13         case '$':return 5;
14         case '(': return 0;
15         case ')': return -1;
16         default: return 8;
17     }
18 }
19 int G(char symbol)
20 {
21     switch(symbol)
22     {
23         case '+':
24         case '-': return 1;
25         case '*':
26         case '/': return 3;
27         case '^':
28         case '$':return 6;
29         case '(': return 9;
30         case ')': return 0;
31         default: return 7;
32     }
33 }
34 }
35 void infix_postfix(char infix[],char postfix[])
36 {
37     int top,i,j;
38     char s[30],symbol;
39     top=-1;
40     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);
}

```

```
Enter the valid infix expression
a+b-c+d
the postfix expression is
ab+c-d+

...Program finished with exit code 0
Press ENTER to exit console.
```

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

```
1  #include <stdio.h>
2  #include<stdlib.h>
3  #define MAX 50
4  void insert();
5  void delete();
6  void display();
7  int queue_array[MAX];
8  int rear = - 1;
9  int front = - 1;
10 int main()
11 {
12     int choice;
13     while (1)
14     {
15         printf("1.Insert element to queue \n");
16         printf("2.Delete element from queue \n");
17         printf("3.Display all elements of queue \n");
18         printf("4.Quit \n");
19         printf("Enter your choice : ");
20         scanf("%d", &choice);
21         switch(choice)
22         {
23             case 1:
24                 insert();
25                 break;
26             case 2:
27                 delete();
28                 break;
29             case 3:
30                 display();
31                 break;
32             case 4:
33                 exit(1);
34             default:
35                 printf("Wrong choice \n");
36         }
37     }
38 }
39 void insert()
40 {
```

```

11 int item;
12 if(rear == MAX - 1)
13 printf("Queue Overflow \n");
14 else
15 {
16 if(front == - 1)
17 front = 0;
18 printf("Inset the element in queue : ");
19 scanf("%d", &item);
20 rear = rear + 1;
21 queue_array[rear] = item;
22 }
23 }
24 void delete()
25 {
26 if(front == - 1 || front > rear)
27 {
28 printf("Queue Underflow \n");
29 return;
30 }
31 else
32 {
33 printf("Element deleted from queue is : %d\n", queue_array[front]);
34 front = front + 1;
35 }
36 }
37 void display()
38 {
39 int i;
40 if(front == - 1)
41 printf("Queue is empty \n");
42 else
43 {
44 printf("Queue is : \n");
45 for(i = front; i <= rear; i++)
46 printf("%d ", queue_array[i]);
47 printf("\n");
48 }
49 }

```

```
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
```

```

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

```

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


```

#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");
    }
}

```

```
49     }
50
51
52     int main()
53     {
54         int choice, f=-1, r=-1, q[max];
55         do
56         {
57
58             printf(" 1: Insert \n 2:Delete \n 3: Display\n 4: Exit\n");
59             printf("Enter your choice\n");
60             scanf("%d", &choice);
61             switch(choice)
62             {
63                 case 1: enqueue(q, &f, &r);
64                     break;
65                 case 2: dequeue(q, &f, &r);
66                     break;
67                 case 3: display( q, &f, &r);
68                     break;
69                 case 4:
70                     break;
71                 default: printf("INVALID CHOICE\n");
72             }
73         }while(choice!=4);
74     }
```



```
✓ ↩ 📄
1: Insert
2:Delete
3: Display
4: Exit
Enter your choice
1
Enter the element:
1
1: Insert
2:Delete
3: Display
4: Exit
Enter your choice
1
Enter the element:
5
1: Insert
2:Delete
3: Display
4: Exit
Enter your choice
1
Enter the element:
3
1: Insert
2:Delete
3: Display
4: Exit
Enter your choice
3
1 6 8
1: Insert
2:Delete
3: Display
4: Exit
Enter your choice
```

```

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
2
4 is deleted
1: Insert
2:Delete
3: Display
4: Exit
Enter your choice
3
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.

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  struct node
5  {
6      int sem;
7      char name[50];
8      char usn[50];
9      struct node *next;
10 };
11 struct node *head= NULL;
12 int c=0;
13 void Insertbegining()
14 {
15     struct node *newnode;
16     int s;
17     char a[50],b[50];
18     printf("Enter your name : ");
19     scanf("%s",a);
20     printf("Enter your usn : ");
21     scanf("%s",b);
22     printf("Enter your semester : ");
23     scanf("%d",&s);
24
25     newnode=(struct node*)malloc(sizeof(struct node));
26     newnode->sem =s;
27     strcpy(newnode->name,a);
28     strcpy(newnode->usn,b);
29
30     newnode->next=head;
31     head=newnode;
32     c++;
33     printf("Node created\n");
34 }
35 void Insertany(int p)
36 {
37     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 *temp1;
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));
100    newnode->sem =s;
101    strcpy(newnode->name,n);
102    strcpy(newnode->usn,u);
103    if (head==NULL)
104    {

```

```

186     head=newnode;
187     printf("first node of linked list created\n");
188     c++;
189 }
190 else
191 {
192     temp=head;
193     while(temp->next!=NULL)
194     {
195         temp=temp->next;
196     }
197     temp->next=newnode;
198     newnode->next=NULL;
199     c++;
200     printf("Node created\n");
201 }
202 }
203 void display()
204 {
205     struct node *ptr;
206     ptr=head;
207     int i=1;
208
209     if(ptr==NULL)
210     {
211         printf("Linked list is empty!\n");
212     }
213     else
214     {
215         while(ptr!= NULL)
216         {
217             printf("----NODE %d----\n",i);
218             printf("Name: %s\n",ptr->name);
219             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();

```

```

        }
    }while(choice!=5);
    return 0;
}

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
```



```
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
Enter your usn : 45678
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 : 4
```

```
----NODE 1----
```

```
Name: rahul
```

```
USN: 12345
```

```
Sem: 3
```

```
----NODE 2----
```

```
Name: rakesh
```

```
USN: 67894
```

```
Sem: 3
```

```
----NODE 3----
```

```
Name: ramesh
```

```
USN: 45678
```

```
Sem: 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

```

1 #include <stdlib.h>
2 #include <string.h>
3 struct node
4 {
5     int sem;
6     struct node *next;
7 };
8 struct node *head= NULL;
9 struct node *head2= NULL;
10 int c=0;
11 void Insert()
12 {
13     struct node *newnode;
14     struct node *temp;
15     int s;
16     printf("Enter integer : ");
17     scanf("%d",&s);
18     newnode=(struct node*)malloc(sizeof(struct node));
19     newnode->sem =s;
20     if (head==NULL)
21     {
22         newnode->next=NULL;
23         head=newnode;
24         printf("first node of linked list created\n");
25         c++;
26     }
27     else
28     {
29         temp=head;
30         while(temp->next!=NULL)
31         {
32             temp=temp->next;
33         }
34         temp->next=newnode;
35         newnode->next=NULL;
36         c++;
37         printf("Node created\n");
38     }
39 }
40 void Insert2()
41 {
42     struct node *newnode;
43     struct node *temp;
44     int s,y;
45     printf("enter elements to create list 2\n");
46     do
47     {

```

```

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;

```

```

        ptr1 = ptr1->next;
        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
```



```

Enter your choice : 1
Enter integer : 70
Node created

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

Enter your choice : 2
before:
 20 50 40 70after:
 20 40 50 70
1. Insert node
2. sort node
3. reverse node
4.concat 2 lists
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 :
0
first node of linked list created

```

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

```

1 #include <stdio.h>
2 #include<stdlib.h>
3 #define MAX_SIZE 5
4
5 int Pque[MAX_SIZE];
6 int n=-1;
7 void enqueue(int);
8 int dequeue();
9 void display();
0 int main(int argc, char **argv)
1- {
2     int option, item;
3     do{
4
5         printf("\n1. Enqueue\n");
6         printf("2. Dequeue\n");
7         printf("3. Display\n");
8         printf("4. Exit\n");
9         printf("Enter the option:");
0         scanf("%d",&option);
1         switch(option)
2         {
3             case 1: printf("\nEnter the item:");
4                     scanf("%d",&item);
5                     enqueue(item);
6                     break;
7             case 2: item=dequeue();
8                     printf("Removed element is : %d\n",item);
9                     break;
0             case 3: display();
1                     break;
2             case 4: exit(0);
3         }
4     }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]);
}

```

Enter the item:1

Enter the item:2

Enter the item:3

Enter the item:4

1. Enqueue
2. Dequeue
3. Display

Enter the option:1

Enter the item:5

1. Enqueue

2. Dequeue

3. Display

4. Exit

Enter the option:1

Enter the item:6

ERROR: Queue is full

1. Enqueue

2. Dequeue

3. Display

4. Exit

Enter the option:3

The Content: 1 2 3 4 5

1. Enqueue

2. Dequeue

3. Display

4. Exit

Enter the option:2

Removed element is : 5

1. Enqueue

2. Dequeue

3. Display

4. Exit

Enter the option:3

The Content: 1 2 3 4

1. Enqueue

2. Dequeue

3. Display

```

The Content: 1 2 3 4 5
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter the option:2
Removed element is : 5

1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter the option:3
The Content: 1 2 3 4
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter the option:2
Removed element is : 4

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

...Program finished with exit code 0

```

Scanned with CamScanner

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


```

1  #include <stdio.h>
2  #include <stdlib.h>
3
4  typedef struct Node {
5      int value;
6      struct Node* prev;
7      struct Node* next;
8  }node;
9
10 node* head = NULL;
11
12 void add_beg(int value) {
13     node* ptr = (node*) malloc(sizeof(node));
14     ptr->value = value;
15     ptr->prev = NULL;
16     ptr->next = head;
17     if(head!=NULL) {
18         head->prev = ptr;
19     }
20     head = ptr;
21 }
22
23 void add_key(int value, int key) {
24     node* tmp = head;
25     while(tmp!=NULL) {
26         if(tmp->value == key) {
27             break;
28         }
29         tmp = tmp->next;
30     }
31     if(tmp==NULL) {
32         printf("\nNo Match\n");
33         return;
34     }
35     if(tmp==head) {
36         add_beg(value);
37         return;
38     }
39     node* ptr = (node*) malloc(sizeof(node));
40     ptr->value = value;
41     ptr->prev = tmp->prev;
42     ptr->next = tmp;
43     (tmp->prev)->next = ptr;
44     tmp->prev = ptr;
45 }
46 void del_key(int key) {
47     if(head == NULL) {
48         printf("\nList is Empty\n");
49         return;
50     }
51     node* tmp = head;
52     while(tmp != NULL) {
53         if(tmp -> value == key) {
54             break;
55         }
56         tmp = tmp->next;
57     }
58     if(tmp==head) {
59         if(head->next==NULL)
60         {
61             free(head);
62             head=NULL;
63             return;
64         }
65         head = head->next;
66         free(head->prev);
67         head->prev = NULL;
68         return;
69     }
70     if(tmp==NULL) {
71         printf("\nNo Match\n");
72         return;
73     }
74     if(tmp->next==NULL) {
75         tmp->prev->next = NULL;

```



```

76     free(tmp);
77     return;
78 }
79 tmp->next->prev = tmp->prev;
80 tmp->prev->next = tmp->next;
81 free(tmp);
82 }
83 void display() {
84     if(head == NULL) {
85         printf("\nList is Empty\n");
86         return;
87     }
88     node* tmp = head;
89     printf("\nLinked list contains : ");
90     while(tmp != NULL) {
91         printf("%d ", tmp->value);
92         tmp = tmp->next;
93     }
94     printf("\n");
95 }
96 void main() {
97     int choice, value, key;
98     while(1) {
99         printf("Enter 1 to add at beginning\n");
100        printf("Enter 2 to add at left of a node\n");
101        printf("Enter 3 to delete a node\n");
102        printf("Enter 4 to display\n");
103        printf("Enter -1 to quit\n");
104        printf("Enter your choice : ");
105        scanf("%d", &choice);
106        if(choice == -1)
107            break;
108        switch(choice) {
109            case 1:
110                printf("\nEnter value to insert : ");
111                scanf("%d", &value);
112                add_beg(value);
113                break;
114            case 2:
115                printf("\nEnter value to insert : ");
116                scanf("%d", &value);
117                printf("\nEnter value of key node : ");
118                scanf("%d", &key);
119                add_key(value, key);
120                break;
121            case 3:
122                printf("\nEnter value of node to be deleted : ");
123                scanf("%d", &key);
124                del_key(key);
125                break;
126            case 4:
127                display();
128                break;
129            default:
130                printf("\n\nWrong Input\n\n");
131        }
132    }
133    printf("\n\n----DONE-----\n\n");
134 }
135

```

```
input
Enter 1 to add at beginning
Enter 2 to add at left of a node
Enter 3 to delete a node
Enter 4 to display
Enter -1 to quit
Enter your choice : 1

Enter value to insert : 4
Enter 1 to add at beginning
Enter 2 to add at left of a node
Enter 3 to delete a node
Enter 4 to display
Enter -1 to quit
Enter your choice : 1

Enter value to insert : 5
Enter 1 to add at beginning
Enter 2 to add at left of a node
Enter 3 to delete a node
Enter 4 to display
Enter -1 to quit
Enter your choice : 2

Enter value to insert : 6

Enter value of key node : 4
Enter 1 to add at beginning
Enter 2 to add at left of a node
Enter 3 to delete a node
Enter 4 to display
Enter -1 to quit
Enter your choice : 4

Linked list contains : 5 6 4
Enter 1 to add at beginning
Enter 2 to add at left of a node

input
Enter -1 to quit
Enter your choice : 2

Enter value to insert : 6

Enter value of key node : 4
Enter 1 to add at beginning
Enter 2 to add at left of a node
Enter 3 to delete a node
Enter 4 to display
Enter -1 to quit
Enter your choice : 4

Linked list contains : 5 6 4
Enter 1 to add at beginning
Enter 2 to add at left of a node
Enter 3 to delete a node
Enter 4 to display
Enter -1 to quit
Enter your choice : 3

Enter value of node to be deleted : 6
Enter 1 to add at beginning
Enter 2 to add at left of a node
Enter 3 to delete a node
Enter 4 to display
Enter -1 to quit
Enter your choice : 4

Linked list contains : 5 4
Enter 1 to add at beginning
Enter 2 to add at left of a node
Enter 3 to delete a node
Enter 4 to display
Enter -1 to quit
Enter your choice : 
```

10. binary tree program

```

1  #include<stdio.h>
2  #include<stdlib.h>
3  struct node
4  {
5      int data;
6      struct node* left;
7      struct node* right;
8  }*root1;
9
10 struct node *create()
11 {
12 {
13     struct node *temp;
14     printf("\n Enter data:");
15     temp=(struct node*)malloc(sizeof(struct node));
16     scanf("%d",&temp->data);
17     temp->left=temp->right=NULL;
18     return temp;
19 }
20
21 void insert(struct node *root,struct node *temp)
22 {
23 {
24     if(temp->data<root->data)
25     {
26     if(root->left!=NULL)

```

```

38     insert(root->left,temp);
39     else
40     root->left=temp;
41     }
42     if(temp->data>root->data)
43     {
44     if(root->right!=NULL)
45     insert(root->right,temp);
46     else
47     root->right=temp;
48     }
49 }
50
51 void Postorder(struct node* node)
52 {
53 {
54     if (node == NULL)
55     return;
56
57     Postorder(node->left);
58     Postorder(node->right);
59     printf("%d ", node->data);
60 }
61 }

```

```

76 void Inorder(struct node* node)
77 {
78     if (node == NULL)
79         return;
80
81     Inorder(node->left);
82
83     printf("%d ", node->data);
84
85     Inorder(node->right);
86 }
87
88 void Preorder(struct node* node)
89 {
90     if (node == NULL)
91         return;
92
93     printf("%d ", node->data);
94
95     Preorder(node->left);
96
97     Preorder(node->right);
98 }
99
100 int main()
101 {
102     int ch;
103     struct node *temp;
104     do
105     {
106         printf("1.create\n2.insert\n3.preorder\n4.postorder\n5.inorder\n6.Exit\n");
107         scanf("%d",&ch);
108         switch(ch)
109         {
110             case 1:
111                 root1=create();
112                 break;
113             case 2:
114                 printf("enter the elem to be entered\n");
115                 temp=(struct node*)malloc(sizeof(struct node));
116                 scanf("%d",&temp->data);
117                 insert(root1,temp);
118                 break;
119             case 3:
120                 Preorder(root1);
121                 printf("\n");
122                 break;
123             case 4:
124                 Postorder(root1);
125                 printf("\n");
126                 break;
127             case 5:
128                 Inorder(root1);
129                 printf("\n");
130                 break;
131             case 6:
132                 break;
133             default:
134                 printf("wrong entry");
135         }
136     }while(ch!=6);
137
138
139
140
141
142
143
144

```

```
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
```