WEEK 6

Lab Exercises:

1: Implement an ascending priority queue.

Note: **An ascending priority queue is a collection of items into which items can be inserted arbitrarily and from which only the smallest item can be removed. If apq is an ascending priority queue, the operation pqinsert(apq,x) inserts element x into apq and pqmindelete(apq) removes the minimum element from apq and returns its value.**

Code:

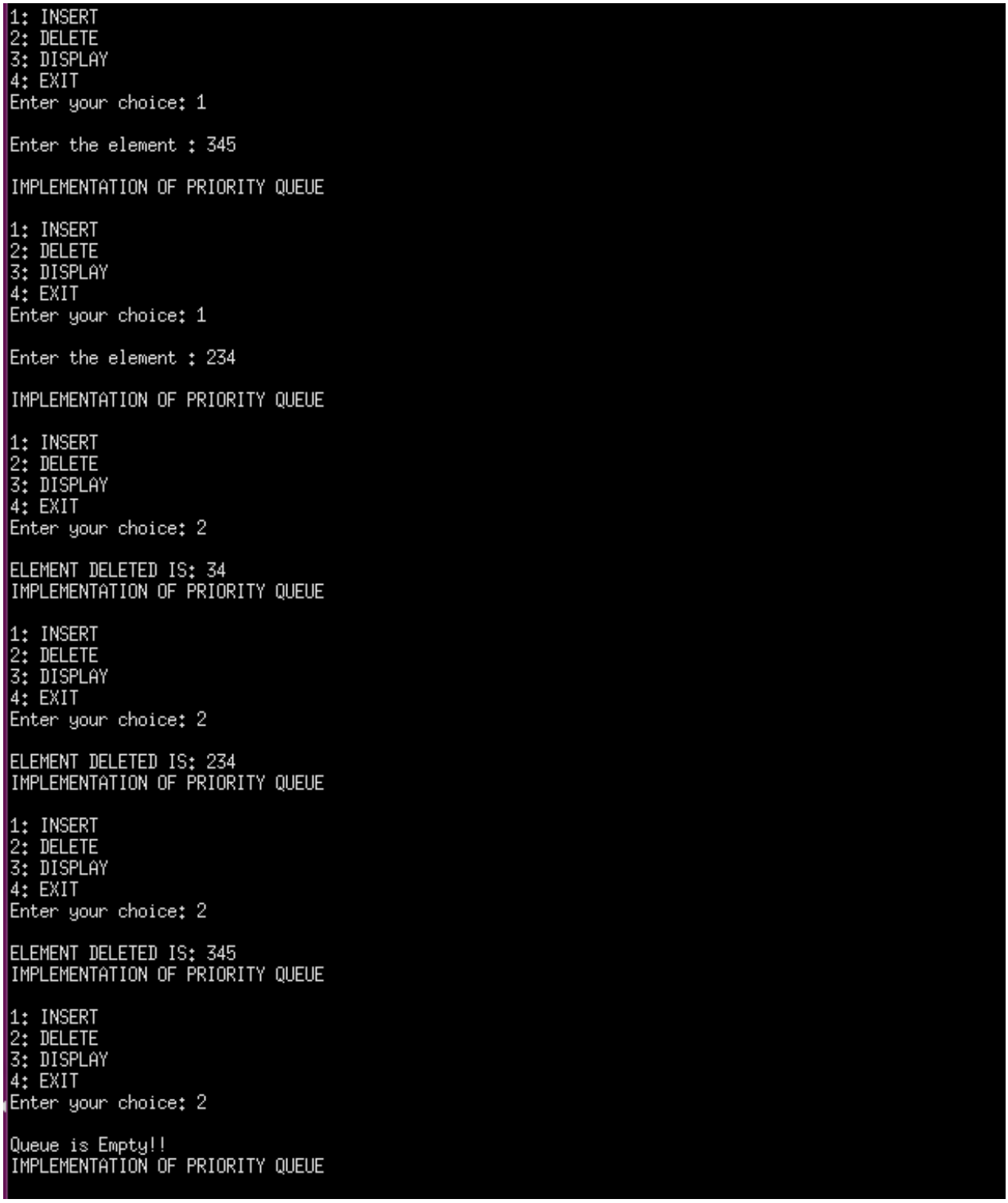
```
#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#define MAXIMUM 45

typedef struct{

int QUEUE[MAXIMUM];

int front;

int rear;

}PQUEUE;

void insertPQ(PQUEUE *queue,int myName){

int i,j;

for(i=queue->front+1;i<=queue->rear;i++)

if(queue->QUEUE[i]>myName)

break;

if(i<=queue->rear){

for(j=queue->rear;j>=i;j--)

queue->QUEUE[j+1]=queue->QUEUE[j];

}
```

```
queue->QUEUE[i]=myName;
queue->rear++;
}
int deletePQ(PQUEUE *queue){
if (queue->front==queue->rear){
printf("\nQueue is Empty!!");
return -1;
}
return queue->QUEUE[++queue->front];
}
void displayPQ(PQUEUE *queue){
if (queue->front==queue->rear){
printf("\nQueue is Empty!!");
return ;
}
int i;
printf("\nQueue is: ");
;
for(i=queue->front+1;i<=queue->rear;i++) printf("\n%d ",queue->QUEUE[i]);
}
int main(){
PQUEUE queue1,*queue;
queue=&queue1;
queue->front=queue->rear=-1;
int myName,i,j;
while(1)
{
int choice;
```

```
printf("\nIMPLEMENTATION OF PRIORITY QUEUE\n");
printf("\n1: INSERT");
printf("\n2: DELETE");
printf("\n3: DISPLAY");
printf("\n4: EXIT");
printf("\nEnter your choice: ");
scanf("%d",&choice);
switch(choice){
case 1:
printf("\nEnter the element : ");
scanf("%d",&myName);
insertPQ(queue,myName);
break;
case 2:
j = deletePQ(queue);
if(j!=-1)printf("\nELEMENT DELETED IS: %d",j);
break;
case 3:
displayPQ(queue);
break;
case 4 :
exit(0);
default:
printf("\nInvalid choice! ");
break;
}
}
return 0;
```

}

Test Case:


1: INSERT
2: DELETE
3: DISPLAY
4: EXIT
Enter your choice: 1
Enter the element : 345
IMPLEMENTATION OF PRIORITY QUEUE
1: INSERT
2: DELETE
3: DISPLAY
4: EXIT
Enter your choice: 1
Enter the element : 234
IMPLEMENTATION OF PRIORITY QUEUE
1: INSERT
2: DELETE
3: DISPLAY
4: EXIT
Enter your choice: 2
ELEMENT DELETED IS: 34
IMPLEMENTATION OF PRIORITY QUEUE
1: INSERT
2: DELETE
3: DISPLAY
4: EXIT
Enter your choice: 2
ELEMENT DELETED IS: 234
IMPLEMENTATION OF PRIORITY QUEUE
1: INSERT
2: DELETE
3: DISPLAY
4: EXIT
Enter your choice: 2
ELEMENT DELETED IS: 345
IMPLEMENTATION OF PRIORITY QUEUE
1: INSERT
2: DELETE
3: DISPLAY
4: EXIT
Enter your choice: 2
Queue is Empty!!
IMPLEMENTATION OF PRIORITY QUEUE

2: Implement a queue of strings using an output restricted dequeue (no deleteRight).
Note: An output-restricted deque is one where insertion can be made at both ends, but deletion can be made from one end only, where as An input-restricted deque is one where deletion can be made from both ends, but insertion can be made at one end only.

Code:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define MAXIMUM 16
#define STRING 26
typedef struct
{
char queue[MAXIMUM][STRING];
int front;
int rear;
} QUEUE;
void insertf(QUEUE *q,char ch[]);
void insertend(QUEUE *q,char ch[]);
void deletef(QUEUE *q);
void display(QUEUE *q);
int main()
{
char myName[STRING],j;
QUEUE *q,q1;
q = &q1;
q1.front = -1;
q1.rear = -1;
while(1)
{
int choice;
printf("\nIMPLEMENTATION OF OUTPUT RESTRICTED DEQUE");
printf("\n1: INSERT FROM FRONT");
printf("\n2: INSERT FRON END");
printf("\n3: DELETE FROM FRONT");
printf("\n4: DISPLAY");
printf("\n5: EXIT");
printf("\nEnter your choice: ");
scanf("%d",&choice);
switch(choice)
{
case 1:
printf("\nEnter the string : ");
scanf("%s",myName);insertf(&q1,myName);
break;
case 2:
printf("\nEnter the string : ");
scanf("%s",myName);
insertend(&q1,myName);
break;
case 3:
printf("\nString deleted is: ");
```

```
deletef(&q1);
break;
case 4:
display(&q1);
break;
case 5:
exit(0);
break;
default :
printf("\nInvalid choice!!");
}
}
return 0;
}
void insertf(QUEUE *q, char ch[])
{
if(q->rear==MAXIMUM-1 && q->front==0)
{
printf("\nQueue is full!!");
return;
}
else if(q->front==-1)
{
q->front=q->rear=0;
strcpy(q->queue[q->front],ch);
}
else if(q->rear!=MAXIMUM-1)
{
int i=q->rear+1;
while(i>q->front)
{
strcpy(q->queue[i],q->queue[i-1]);
i=i-1;
}
strcpy(q->queue[q->front],ch);
q->rear++;
}
else
{
q->front--;
strcpy(q->queue[q->front],ch);
}
}
void insertend(QUEUE *q, char ch[])
{
if(q->rear==MAXIMUM-1 && q->front==0)
{
printf("\nQueue is full!!");
return;
}
else if(q->front==-1)
{
```

```
q->front=q->rear=0;
strcpy(q->queue[q->rear],ch);
}
else if(q->front!=0)
{
int i=q->front-1;
while(i<q->rear)
{
strcpy(q->queue[i],q->queue[i+1]);
i=i+1;
}
strcpy(q->queue[q->rear],ch);
q->front--;
}
else
{
q->rear++;
strcpy(q->queue[q->rear],ch);
}
}
void deletef(QUEUE *q)
{
if(q->front==-1)
{
printf("\nQueue is empty!!");
return ;
}
char ch[STRING] ;
strcpy(ch, q->queue[q->front]);
if(q->front==q->rear)
q->front=q->rear=-1;
else
q->front++;
printf("\n%s \n",ch);
}void display(QUEUE *q)
{
if(q->front==-1)
{
printf("\nQueue is empty!!");
return ;
}
int i;
printf("\nQUEUE IS: ");
for(i=q->front; i!=q->rear; i++)
{
printf("\n%s ",q->queue[i]);
}
printf("\n%s ",q->queue[i]);
}
```

Test Case:

```
IMPLEMENTATION OF OUTPUT RESTRICTED DEQUE
1: INSERT FROM FRONT
2: INSERT FRON END
3: DELETE FROM FRONT
4: DISPLAY
5: EXIT
Enter your choice: 1

Enter the string : Danish

IMPLEMENTATION OF OUTPUT RESTRICTED DEQUE
1: INSERT FROM FRONT
2: INSERT FRON END
3: DELETE FROM FRONT
4: DISPLAY
5: EXIT
Enter your choice: 2

Enter the string : Eqbal

IMPLEMENTATION OF OUTPUT RESTRICTED DEQUE
1: INSERT FROM FRONT
2: INSERT FRON END
3: DELETE FROM FRONT
4: DISPLAY
5: EXIT
Enter your choice: 3

String deleted is:
Danish

IMPLEMENTATION OF OUTPUT RESTRICTED DEQUE
1: INSERT FROM FRONT
2: INSERT FRON END
3: DELETE FROM FRONT
4: DISPLAY
5: EXIT
Enter your choice: 4

QUEUE IS:
Eqbal
IMPLEMENTATION OF OUTPUT RESTRICTED DEQUE
1: INSERT FROM FRONT
2: INSERT FRON END
3: DELETE FROM FRONT
4: DISPLAY
5: EXIT
```

3: Write a program to check whether given string is a palindrome using a dequeue.

Code:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define MAXIMUM 50
typedef struct
{
int queue[MAXIMUM];
int front;
int rear;
} QUEUE;
void insertf(QUEUE *q,char c);
char deleteend(QUEUE *q);
char deletef(QUEUE *q);
void display(QUEUE *q);
int main()
{
int i,j;
QUEUE *q,q1;
q = &q1;
q1.front = -1;
q1.rear = -1;
char a,b;
int s;
char ch[20];
printf("\nPROGRAM TO CHECK WHETHER GIVEN STRING IS A PALINDROME OR NOT USING A DEQUEUE");
printf("\nEnter the string to be checked: ");
scanf("%s",ch);
for(s=0; ch[s]!='\0'; s++)
{
insertf(q,ch[s]);
}
for(s=0; s<strlen(ch)/2; s++)
{
a=deleteend(q);
b=deletef(q);
if(a!=b)
{
printf("\nEntered string is not a palindrome.");
exit(0);
}
}
printf("\nEntered string is a palindrome");
return 0;
}
void insertf(QUEUE *q, char ch)
{
if(q->rear==MAXIMUM-1 && q->front==0)
{
```

```
printf("\nQueue is full FULL!!");
return;}
else if(q->front==-1)
{
q->front=q->rear=0;
q->queue[q->front]=ch;
}
else if(q->rear!=MAXIMUM-1)
{
int s=q->rear+1;
while(s>q->front)
{
q->queue[s]=q->queue[s-1];
s=s-1;
}
q->queue[q->front]=ch;
q->rear++;
}
else
{
q->front--;
q->queue[q->front]=ch;
}
}
char deleteend(QUEUE *q)
{
if(q->front==-1)
{
printf("EMPTY\n");
return -1;
}
int val = q->queue[q->rear];
if(q->front==q->rear)
q->front=q->rear=-1;
else
q->rear--;
return val;
}
char deletef(QUEUE *q)
{
if(q->front==-1)
{
printf("\nQueue is empty!!");
return -1;
}
int val = q->queue[q->front];
if(q->front==q->rear)
q->front=q->rear=-1;
else
q->front++;
return val;
}
```

```
void display(QUEUE *q)
{if(q->front==-1)
{
printf("\nQueue is empty!!");
return ;
}
int i;
printf("\nQUEUE IS: ");
for(i=q->front; i!=q->rear; i++)
{
printf("\n%d ",q->queue[i]);
}
printf("\n%d ",q->queue[i]);
}
```

Test Case:

```
PROGRAM TO CHECK WHETHER GIVEN STRING IS A PALINDROME OR NOT USING A DEQUEUE
Enter the string to be checked: Danish

Entered string is not a palindrome.
Process returned 0 (0x0)   execution time : 5.338 s
Press ENTER to continue.
```

```
PROGRAM TO CHECK WHETHER GIVEN STRING IS A PALINDROME OR NOT USING A DEQUEUE
Enter the string to be checked: MOM

Entered string is a palindrome
Process returned 0 (0x0)   execution time : 5.429 s
Press ENTER to continue.
```