Manoj M Mallya

200905130

Section C2

Roll no 23

LAB – 6

Example question :

1) Implement a dequeue of integers with following functions.

a) deleteLeft b) addLeft c) deleteRight d) addRight e) display

CODE :

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 30
typedef struct dequeue
{
int data[MAX];
int rear,front;
}dequeue;
void initialize(dequeue *p);
int empty(dequeue *p);
int full(dequeue *p);
void enqueueR(dequeue *p,int x);
void enqueueF(dequeue *p,int x);
int dequeueF(dequeue *p);
```

```
int dequeueR(dequeue *p);
void print(dequeue *p);
void initialize(dequeue *P)
{
 P->rear=-1;
 P->front=-1;
}
int empty(dequeue *P)
{
 if(P->rear==-1)
 return(1);
 return(0);
}
int full(dequeue *P)
{
 if((P->rear+1)%MAX==P->front)
 return(1);
 return(0);
}
void enqueueR(dequeue *P,int x)
{
 if(empty(P))
 {
 P->rear=0;
 P->front=0;
 P->data[0]=x;
```

```
 }
 else
 {
 P->rear=(P->rear+1)%MAX;
 P->data[P->rear]=x;
 }
}
void enqueueF(dequeue *P,int x)
{
 if(empty(P))
 {
 P->rear=0;
 P->front=0;
 P->data[0]=x;
 }
 else
 {
 P->front=(P->front-1+MAX)%MAX;
 P->data[P->front]=x;
 }
}
int dequeueF(dequeue *P)
{
 int x;
 x=P->data[P->front];
 if(P->rear==P->front)
```

```
 /*delete the last element */
 initialize(P);
 else
 P->front=(P->front+1)%MAX;
 return(x);
}
int dequeueR(dequeue *P)
{
 int x;
 x=P->data[P->rear];
 if(P->rear==P->front)
 initialize(P);
 else
 P->rear=(P->rear-1+MAX)%MAX;
 return(x);
}
void print(dequeue *P)
{
 if(empty(P))
 {
 printf("\nQueue is empty!!");exit(0);
 }
 int i;
 i=P->front;
 while(i!=P->rear)
 {
```

```
 printf("\n%d",P->data[i]);
 i=(i+1)%MAX;
 }
 printf("\n%d\n",P->data[P->rear]);
}
void main()
{
 printf("Manoj M Mallya\n 200905130\n SECTION C2\n ROLL NO. 23\n\n\n");
 int i,x,op,n;
 dequeue q;
 initialize(&q);
 do
 {

printf("\n1.Create\n2.Insert(rear)\n3.Insert(front)\n4.Delete(rear)\n5.Delete(fr
ont)");
 printf("\n6.Print\n7.Exit\n\nEnter your choice:");
 scanf("%d",&op);
 switch(op)
 {

 case 1: printf("\nEnter number of elements:");
 scanf("%d",&n);
 initialize(&q);printf("\nEnter the data:");
 for(i=0;i<n;i++)
 {
 scanf("%d",&x);
```

```
if(full(&q))
{
printf("\nQueue is full!!");
exit(0);
}
enqueueR(&q,x);
}
break;

case 2: printf("\nEnter element to be inserted:");
scanf("%d",&x);
if(full(&q))
{
printf("\nQueue is full!!");
exit(0);
}
enqueueR(&q,x);
break;

case 3: printf("\nEnter the element to be inserted:");
scanf("%d",&x);
if(full(&q))
{
printf("\nQueue is full!!");
exit(0);
}
```

```
enqueueF(&q,x);
break;

case 4:
if(empty(&q))
{
printf("\nQueue is empty!!");
exit(0);
}
x=dequeueR(&q);
printf("\nElement deleted is %d\n",x);
break;

case 5:
if(empty(&q))
{
printf("\nQueue is empty!!");
exit(0);
}
x=dequeueF(&q);
printf("\nElement deleted is %d\n",x);
break;

case 6: print(&q);
break;
```

```
 default: break;
 }
 }while(op!=7);
}
```

OUTPUT :

```
Manoj M Mallya
 200905130
 SECTION C2
 ROLL NO. 23



1.Create
2.Insert(rear)
3.Insert(front)
4.Delete(rear)
5.Delete(front)
6.Print
7.Exit

Enter your choice:1

Enter number of elements:4

Enter the data:2 4 8 16

1.Create
2.Insert(rear)
3.Insert(front)
4.Delete(rear)
5.Delete(front)
6.Print
7.Exit

Enter your choice:2

Enter element to be inserted:32

1.Create
2.Insert(rear)
3.Insert(front)
4.Delete(rear)
5.Delete(front)
6.Print
7.Exit

Enter your choice:3
```

```
Enter your choice:3

Enter the element to be inserted:1

1.Create
2.Insert(rear)
3.Insert(front)
4.Delete(rear)
5.Delete(front)
6.Print
7.Exit

Enter your choice:6

1
2
4
8
16
32

1.Create
2.Insert(rear)
3.Insert(front)
4.Delete(rear)
5.Delete(front)
6.Print
7.Exit

Enter your choice:4

Element deleted is 32

1.Create
2.Insert(rear)
3.Insert(front)
4.Delete(rear)
5.Delete(front)
6.Print
7.Exit

Enter your choice:5
```

```
Enter your choice:5

Element deleted is 1

1.Create
2.Insert(rear)
3.Insert(front)
4.Delete(rear)
5.Delete(front)
6.Print
7.Exit

Enter your choice:6

2
4
8
16

1.Create
2.Insert(rear)
3.Insert(front)
4.Delete(rear)
5.Delete(front)
6.Print
7.Exit

Enter your choice:7

Process returned 7 (0x7)   execution time : 53.539 s
Press any key to continue.
```

Exercise Questions :

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>

#define MAX_QUEUE_SIZE 10
```

```
typedef struct
{
    int front, rear;
    int array[MAX_QUEUE_SIZE];
} Queue;
void display(Queue q)
{
    if (q.front == -1 && q.rear == -1)
    {
        printf("\nThe queue is empty.");
    }
    else
    {
        printf("\n");
        for (int i = q.front; i <= q.rear; i++)
        {
            printf(" %3d ", q.array[i]);
        }
    }
}
void push(Queue *q, int key)
{
    if(q->rear == MAX_QUEUE_SIZE - 1)
        printf("\nThe queue is full");
    else
    {
```

```
        if (q->front == -1 && q->rear == -1)
            q->front++;
        int pos;
        for(int i = q->front; i<=q->rear; i++)
        {
            if(q->array[i]<=key)
                pos = i+1;
        }
        for(int i = q->rear; i>=pos; i--)
            q->array[i+1] = q->array[i];
        q->rear++;
        q->array[pos] = key;
    }
}
int pop(Queue *q)
{
    int temp = q->array[q->front];
    q->front++;
    if (q->front > q->rear)
    {
        q->front = -1;
        q->rear = -1;
    }
    return temp;
}
void main()
```

```
{
    printf("Manoj M Mallya\n 200905130\n SECTION C2\n ROLL NO. 23\n\n\n");
    Queue q;
    q.front = -1;
    q.rear = -1;
    int choice = 0, ele;
    do
    {
        printf("\n1: Display the Queue \n2: Pop \n3: Push an element \n4: Exit");
        printf("\nEnter the operation to be done: ");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            display(q);
            break;

        case 2:
            if (q.front == -1 && q.rear == -1)
                printf("\nThe queue is empty");
            else
            {
                ele = pop(&q);
                printf("\nElement poppped is %d", ele);
            }
            break;
```

```
        case 3:
            printf("\nEnter the element to be pushed : ");
            scanf("%d", &ele);
            push(&q, ele);
            break;
        }
        printf("\n");
    }
    while(choice!=4);
}
```

OUTPUT :

```
Manoj M Mallya
 200905130
 SECTION C2
 ROLL NO. 23



1: Display the Queue
2: Pop
3: Push an element
4: Exit
Enter the operation to be done: 1

The queue is empty.

1: Display the Queue
2: Pop
3: Push an element
4: Exit
Enter the operation to be done: 3

Enter the element to be pushed : 43


1: Display the Queue
2: Pop
3: Push an element
4: Exit
Enter the operation to be done: 3

Enter the element to be pushed : 29


1: Display the Queue
2: Pop
3: Push an element
4: Exit
Enter the operation to be done: 3

Enter the element to be pushed : 64
```

```
Enter the element to be pushed : 64


1: Display the Queue
2: Pop
3: Push an element
4: Exit
Enter the operation to be done: 1

  29   43   64

1: Display the Queue
2: Pop
3: Push an element
4: Exit
Enter the operation to be done: 2

Element poppped is 29

1: Display the Queue
2: Pop
3: Push an element
4: Exit
Enter the operation to be done: 2

Element poppped is 43

1: Display the Queue
2: Pop
3: Push an element
4: Exit
Enter the operation to be done: 2

Element poppped is 64
```

```
Enter the operation to be done: 2

Element poppped is 64

1: Display the Queue
2: Pop
3: Push an element
4: Exit
Enter the operation to be done: 2

The queue is empty

1: Display the Queue
2: Pop
3: Push an element
4: Exit
Enter the operation to be done: 1

The queue is empty.

1: Display the Queue
2: Pop
3: Push an element
4: Exit
Enter the operation to be done: 4


Process returned 4 (0x4)   execution time : 75.567 s
Press any key to continue.
```

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>

#define MAX_QUEUE_SIZE 5

typedef struct

{

   int front, rear;
```

```
    char* array[MAX_QUEUE_SIZE];
} Queue;
void print(Queue q)
{
    if (q.front == -1 && q.rear == -1)
        printf("\nThe queue is empty");
    else
    {
        printf("\n");
        for (int i = q.front; i <= q.rear; i++)
            printf("%s\t", q.array[i]);
    }
}
void pushRight(Queue *q, char* key)
{
    if (q->rear == MAX_QUEUE_SIZE - 1)
        printf("\nThe queue is full");
    else
    {
        if (q->front == -1 && q->rear == -1)
            q->front++;
        q->array[++q->rear] = key;
    }
}
void pushLeft(Queue *q, char* key)
{
```

```
    if (q->rear == MAX_QUEUE_SIZE - 1)
        printf("\nThe queue is full");
    else
    {
        if (q->front == -1 && q->rear == -1)
            q->front++;
        for(int i = q->rear; i>=q->front; i--)
            q->array[i+1] = q->array[i];
        ++q->rear;
        q->array[q->front] = key;
    }
}
char* pop(Queue *q)
{
    char* temp = q->array[q->front];
    q->front++;
    if (q->front > q->rear)
    {
        q->front = -1;
        q->rear = -1;
    }
    return temp;
}
char* front(Queue q)
{
    return q.array[q.front];
```

```
}
void main()
{
   printf("Manoj M Mallya\n 200905130\n SECTION C2\n ROLL NO. 23\n\n\n");
   Queue q;
   q.front = -1;
   q.rear = -1;
   int ch = 0;
   char* ele;
   do
   {
      printf("\n1 : Display the Queue \n2 : Pop \n3 : Push element from Right\n4 : Push element from Left\n5 : Exit");
      printf("\nEnter the operation to be done: ");
      scanf("%d", &ch);
      switch(ch)
      {
      case 1:
         print(q);
         break;
      case 2:
         if (q.front == -1 && q.rear == -1)
            printf("\nThe queue is empty");
         else
         {
            ele = pop(&q);
```

```
            printf("\nElement popped is %s", ele);
        }
        break;
    case 3:
        ele = (char*)calloc(100, sizeof(char));
        printf("\nEnter the element : ");
        scanf(" %s", ele);
        pushRight(&q, ele);
        break;
    case 4:
        ele = (char*)calloc(100, sizeof(char));
        printf("\nEnter the element : ");
        scanf(" %s", ele);
        pushLeft(&q, ele);
        break;
    }
    printf("\n");
  }
  while(ch!=5);
}
```

OUTPUT :

```
Manoj M Mallya
 200905130
 SECTION C2
 ROLL NO. 23



1 : Display the Queue
2 : Pop
3 : Push element from Right
4 : Push element from Left
5 : Exit
Enter the operation to be done: 1

The queue is empty

1 : Display the Queue
2 : Pop
3 : Push element from Right
4 : Push element from Left
5 : Exit
Enter the operation to be done: 3

Enter the element : My


1 : Display the Queue
2 : Pop
3 : Push element from Right
4 : Push element from Left
5 : Exit
Enter the operation to be done: 3

Enter the element : name


1 : Display the Queue
2 : Pop
3 : Push element from Right
4 : Push element from Left
5 : Exit
Enter the operation to be done: 3
```

```
Enter the operation to be done: 3

Enter the element : is


1 : Display the Queue
2 : Pop
3 : Push element from Right
4 : Push element from Left
5 : Exit
Enter the operation to be done: 3

Enter the element : Manoj


1 : Display the Queue
2 : Pop
3 : Push element from Right
4 : Push element from Left
5 : Exit
Enter the operation to be done: 1

My      name    is      Manoj

1 : Display the Queue
2 : Pop
3 : Push element from Right
4 : Push element from Left
5 : Exit
Enter the operation to be done: 4

Enter the element : Hello,


1 : Display the Queue
2 : Pop
3 : Push element from Right
4 : Push element from Left
5 : Exit
Enter the operation to be done: 1

Hello,  My      name    is      Manoj
```

```
1 : Display the Queue
2 : Pop
3 : Push element from Right
4 : Push element from Left
5 : Exit
Enter the operation to be done: 3

Enter the element : bye

The queue is full

1 : Display the Queue
2 : Pop
3 : Push element from Right
4 : Push element from Left
5 : Exit
Enter the operation to be done: 2

Element popped is Hello,

1 : Display the Queue
2 : Pop
3 : Push element from Right
4 : Push element from Left
5 : Exit
Enter the operation to be done: 2

Element popped is My

1 : Display the Queue
2 : Pop
3 : Push element from Right
4 : Push element from Left
5 : Exit
Enter the operation to be done: 2

Element popped is name
```

```
1 : Display the Queue
2 : Pop
3 : Push element from Right
4 : Push element from Left
5 : Exit
Enter the operation to be done: 2

Element popped is is

1 : Display the Queue
2 : Pop
3 : Push element from Right
4 : Push element from Left
5 : Exit
Enter the operation to be done: 2

Element popped is Manoj

1 : Display the Queue
2 : Pop
3 : Push element from Right
4 : Push element from Left
5 : Exit
Enter the operation to be done: 2

The queue is empty

1 : Display the Queue
2 : Pop
3 : Push element from Right
4 : Push element from Left
5 : Exit
Enter the operation to be done: 5


Process returned 5 (0x5)   execution time : 124.533 s
Press any key to continue.
```

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

CODE :

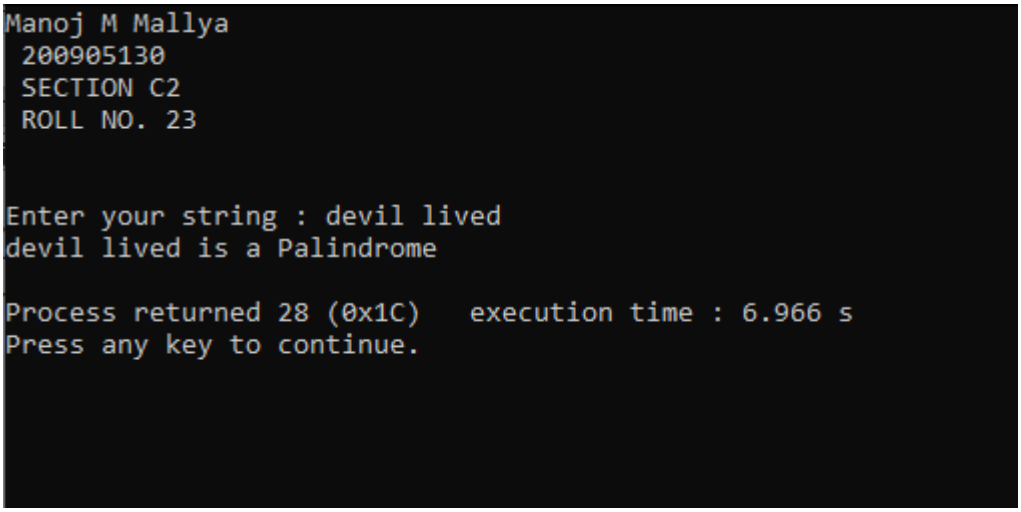
```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX_QUEUE_SIZE 15

typedef struct

{
```

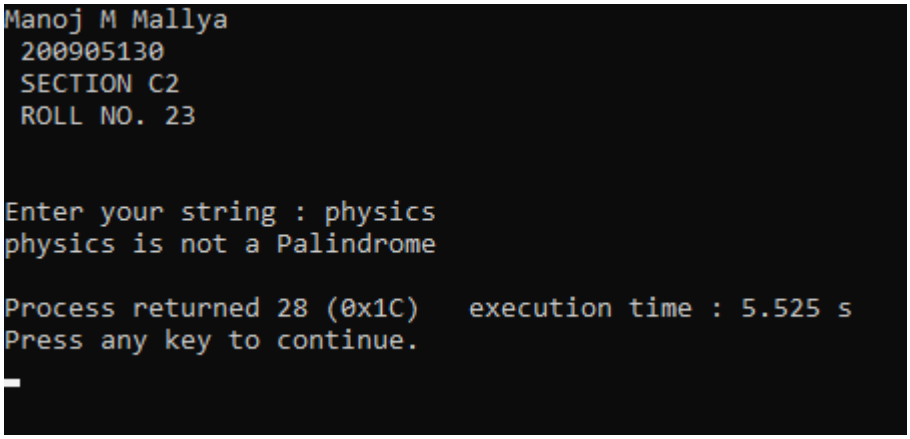
```
    int front, rear;
    char array[MAX_QUEUE_SIZE];
} Queue;
void pushR(Queue *q, char key)
{
    if (q->rear == MAX_QUEUE_SIZE - 1)
        printf("\nThe queue is full");
    else
    {
        if (q->front == -1 && q->rear == -1)
            q->front++;
        q->array[++q->rear] = key;
    }
}
char popRight(Queue *q)
{
    char temp = q->array[q->rear];
    q->rear--;
    if (q->front > q->rear)
    {
        q->front = -1;
        q->rear = -1;
    }
    return temp;
}
char popLeft(Queue *q)
```

```
{
    char temp = q->array[q->front];
    q->front++;
    if (q->front > q->rear)
    {
        q->front = -1;
        q->rear = -1;
    }
    return temp;
}
void main()
{
    printf("Manoj M Mallya\n 200905130\n SECTION C2\n ROLL NO. 23\n\n\n");
    Queue q;
    q.front = q.rear = -1;
    char ele[100];
    printf("Enter your string : ");
    gets(ele);
    int n = strlen(ele);
    for(int i = 0; i<n; i++)
        pushR(&q, ele[i]);
    n = n/2;
    int p = 1;
    while(n--)
    {
        if(popLeft(&q)!=popRight(&q))
```

```
        {
            p = 0;
            break;
        }
    }
    if(p)
        printf("%s is a Palindrome\n",ele);
    else
        printf("%s is not a Palindrome\n",ele);
}
```

OUTPUT :




Manoj M Mallya
200905130
SECTION C2
ROLL NO. 23
Enter your string : physics
physics is not a Palindrome
Process returned 28 (0x1C) execution time : 5.525 s
Press any key to continue.