28.10.2020

1) WAP to implement a double ended Queue (dequeue)

```
#include <stdio.h>
#define qsize 5
int f=0, r=-1, ch;
int item, q[10];
int isfull()
 {
return (r==qsize-1)?1:0;
}

int isempty()
{
return (f>r)?1:0;
}

void insert_rear()
{
```

```
if (isfull())
 {
 printf ("queue overflow\n");
return;
 }
r=r+1;
q[r] = item;
}
void delete_front()
 {
if (isempty())
 {
 printf ("queue is empty\n");
return;
}
printf ("item deleted is %d \n ", q[(f)++]);
 if (f>r)
    {
  f=0; r=-1;
   }
}

void insert_front()
{
```

```
if (f!=0)
  {
  f=f-1;
  q[f]=item;
  return;
  }
else if ((f==0)&&(r==-1))
{
q[++r]=item;
return;
}
else
printf("insertion not possible\n");
}

void delete_rear()
 {
 if (isempty())
  {
 printf("queue is empty\n");
 return;
 }
 printf("item deleted is %d\n", q[(r)--]);
 if (f>r)
```

```
{
f=0;
r=-1;
}
}

void display()
{
int i;
if(isempty())
{
printf("queue is empty\n");
return;
}
for(i=f; i<=r; i++)
printf("%d\n", q[i]);
}

int main()
{
do
{
printf("1. insert_rear\n2. insert_front\n3. delete_rear\n4. delete_front\n
5. display\n6. exit\n");
```

```
printf("enter choice\n");
scanf("%d", &ch);
switch(ch)
{
case 1 : printf("enter the item\n");
         scanf("%d", &item);
         insert_rear();
         break;
case 2 : printf("enter the item\n");
         scanf("%d", &item);
         insert_front();
         break;
case 3 : delete_rear();
         break;
case 4 : delete_front();
         break;
case 5 : display();
         break;
default : break;
  }
}
while(ch!=6)
return 0;
}
```

2) WAP to implement input restricted dequeue

```
#include <stdio.h>
#define qsize 5
int f=0, r=-1, ch;
int item, q[10];
int isfull()
{
return (r==qsize-1)?1:0;
}
int isempty()
{
return (f>r)?1:0;
}

void insert_rear()
{
if(isfull())
{
printf("queue overflow\n");
return;
}
r=r+1;
q[r]=item;
```

```
}

void delete_front()
{
 if (isempty())
  {
 printf("queue empty\n");
 return;
 }
printf("item deleted is %d\n", q[(f)++]);
 if (f>r)
  {
  f=0; r=-1;
   }
  }


void delete_rear()
{
 if (isempty())
  {
 printf("queue is empty\n");
 return;
}
printf("item deleted is %d\n", q[(r)--]);
```

```
if (f>r)
{
f=0; r=-1;
 }
}

void display ()
{
 int i;
if (isempty())
{
printf ("queue empty \n");
return;
}
for (i=f; i<=r; i++)
   printf ("%d\n", q[i]);
}

int main ()
{
printf ("1. insert_rear \n 2. delete_~~front~~ rear \n 3. delete_front \n 4. display
   \n 5. exit \n");
do
{
```

```
printf("enter choice\n");
scanf("%d", &ch);
switch(ch)
{
case 1 : printf("enter the item\n");
         scanf("%d", &item);
         insert_rear();
         break;
case 2 : delete_rear();
         break;
case 3 : delete_front();
         break;
case 4 : display();
         break;
default : break;
  }
}
while(ch!=5);
return 0;
}
```

3) WAP to implement output restricted dequeue

```
#include <stdio.h>
#define qsize 5
int f=0, r=-1, ch;
int item, q[10];
int isfull()
{
return (r==qsize-1)?1:0;
}

int isempty()
{
return (f>r)?1:0;
}

void insert_rear()
{
if (isfull())
{
printf("queue overflow\n");
return;
}
r=r+1;
```

```
q[r] = item;
}

void delete_front()
{
if (isempty())
{
printf("queue empty \n");
return;
}
printf("item deleted is %d\n", q[f]++);
if(f>r)
{
f=0; r=-1;
}
}

void insert_front()
{
if(f!=0)
{
f=f-1;
q[f] = item;
return;
```

```
}
else if ((f==0) && (r==-1))
 {
  q[++r]=item;
  return;
 }
else
printf("insertion not possible.\n");
 }

void display()
{
int i;
if (isempty())
 {
printf("queue empty\n");
 return;
}
for (i=f; i<=r; i++)
printf("%d\n", q[i]);
}

int main()
{
```

```
printf ("1. insert_rear\n 2. insert_front\n 3. delete_front\n 4.
   display\n 5. exit\n");
do
{
printf ("enter choice\n");
scanf ("%d", &ch);
switch (ch)
{
case 1 : printf ("enter the item\n");
         scanf ("%d", &item);
         insert_rear();
         break;
case 2 : printf ("enter the item\n")
         scanf ("%d", &item);
         insert_front();
          break;
case 3 : delete_front();
         break;
case 4 : display();
         break;
default : break;
}
}
while (ch!=5);
return 0;
}
```