**Experiment 13**

**Deque Implementation Using Array**

**Date :** 05-10-2020

**Aim:** To implement a Deque using array

**Data Structure used :** Deque, Array

**Algorithms**

**1. Algorithm for insertion in Front**

Input: An Array implementation of Deque (DQ[SIZE]), with front pointing to the first element and rear pointing to the last element in and an element E to be inserted into the queue.

Output: The Deque with the element E inserted at the front

Data Structure: Deque

Steps:

Step 1: if(front == 0) then

Step 1: print(“The insertion at front is not possible”)

Step 2: exit(1)

Step 2: else

Step 1: if(rear == -1) then //If there is no element in the deque then insetion at

Step 1: front ++ //front is possible

Step 2: else

Step 1: front - -

Step 3: EndIf

Step 4: DQ[front] = E

Step 3: EndIf

**2. Algorithm for insertion in Rear**

Input: An Array implementation of Deque (DQ[SIZE]), with front pointing to the first element and rear pointing to the last element in and an element E to be inserted into the queue.

Output: The Deque with the element E inserted at the rear

Data Structure: Deque

Steps:

Step 1: if(rear == SIZE) then

Step 1: print(“The queue is full insertion not possible”)

Step 2: exit(1)

Step 2: else

Step 1: if(rear == -1) then

Step 1: front ++

Step 2: EndIf

Step 3: DQ[++rear] = E

Step 3: EndIf

**3. Algorithm for removing from front**

Input: An Array implementation of Deque (DQ[SIZE]), with front pointing to the first element and rear pointing to the last element in the queue.

Output: The element E which is removed form the front of the deque

Data Structure: Deque

Steps

Step 1: if(front == -1) then

Step 1: print(“The Deque is empty”)

Step 2: exit(1)

Step 2: else

Step 1: E = DQ[front]

Step 2: if(front == rear) then

Step 1: front =-1

Step 2: rear =-1

Step 3: else

Step 1: front--

Step 4: endif

Step 3: endif

**4. Algorithm for removing from the rear**

Input: An Array implementation of Deque (DQ[SIZE]), with front pointing to the first element and rear pointing to the last element in the queue.

Output: The element E which is removed form the rear of the deque

Data Structure: Deque

Steps

Step 1: if(rear == -1) then

Step 1: print(“The Deque is empty”)

Step 2: exit(1)

Step 2: else

Step 1: E = DQ[rear]

Step 2: if(front == rear) then

Step 1: front =-1

Step 2: rear =-1

Step 3: else

Step 1: rear - -

Step 4: endif

Step 3: endif

**Program code:**

/\* Deque implemetation using dynamic array

\* Done By : Rohit Karuankaran

\* \*/

#include <stdlib.h>

#include <stdio.h>

#define SIZE 50

typedef struct deque\_structure\_datatype

{

int \*Q;

int size;

int front;

int rear;

}deque;

void initQueue(deque \*dq)

{

dq->size = SIZE;

dq->Q = (int\*) malloc(dq->size\*sizeof(int));

dq->front = -1;

dq->rear = -1;

}

void delQueue(deque \*dq)

{

free(dq->Q);

}

void insertRear(deque \*dq,int elem)

{

if(dq->rear>=dq->size)

{

printf("The Queue is full Inseriton not possible\n");

//incrSize(dq);

}

else

{

if(dq->front==-1)

{

dq->front=dq->front+1;

}

dq->rear = dq->rear+1;

dq->Q[dq->rear] = elem;

return;

}

}

void insertFront(deque \*dq,int elem)

{

if(dq->front==0)

{

//This is the condition if there is somthin inserted

printf("Insertion at front not possible\n");

}

else

{

if(dq->rear == -1)

{

dq->rear= dq->rear+1;

}

if(dq->front == -1)

{

dq->front=dq->front+1;

}

else

{

dq->front = dq->front-1;

}

dq->Q[dq->front] = elem;

return;

}

}

int deleteFront(deque \*dq)

{

if(dq->front == -1)

{

printf("QUEUE IS EMPTY THERE IS NO ELEMENT TO DELETE\n");

return -1;

}

else

{

int elem = dq->Q[dq->front];

if(dq->front==dq->rear)

{

dq->front = -1;

dq->rear = -1;

}

else

dq->front=dq->front+1;

return elem;

}

}

int deleteRear(deque \*dq)

{

if(dq->rear ==-1)

{

printf("QUEUE IS EMPTY THERE IS NO ELEMENT TO DELETE\n");

return -1;

}

else

{

int elem = dq->Q[dq->rear];

if(dq->front==dq->rear)

{

dq->front = -1;

dq->rear = -1;

}

else

{

dq->rear = dq->rear-1;

}

return elem;

}

}

void displayQueue(deque \*dq)

{

int i = dq->front;

if(dq->front)

{

printf("EMPTY");

return;

}

while(i>=0&&i<=dq->rear)

{

printf("%d ",dq->Q[i]);

i++;

}

}

int main()

{

deque \*myDeque = (deque\*) malloc(sizeof(deque));

int RUN = 1;

int elem;

int choice;

initQueue(myDeque);

while(RUN)

{

printf("\n=======================\n");

printf(" Menu\n");

printf("=======================\n");

printf("1.Enter into the front\n");

printf("2.Enter into the rear\n");

printf("3.Remove from the front\n");

printf("4.Remove from the rear\n");

printf("5.Display the deque\n");

printf("6.Exit\n");

printf("Enter your choice : ");

scanf("%d%\*c",&choice);

switch(choice)

{

case 1: printf("Enter the element you want to enter into the front : ");

scanf("%d%\*c",&elem);

insertFront(myDeque,elem);

break;

case 2: printf("Enter the element you want to enter into the rear: ");

scanf("%d%\*c",&elem);

insertRear(myDeque,elem);

break;

case 3: elem = deleteFront(myDeque);

printf("The element remove is :%d\n",elem);

break;

case 4: elem = deleteRear(myDeque);

printf("The element remove is :%d\n",elem);

break;

case 5: printf("The Queue is: ");

displayQueue(myDeque);

printf("\n");

break;

case 6: RUN = 0;

break;

default: printf("Enter a valid input\n\n");

}

}

/\*

insert(myDeque,32);

insert(myDeque,21);

displayQueue(myDeque);

\*/

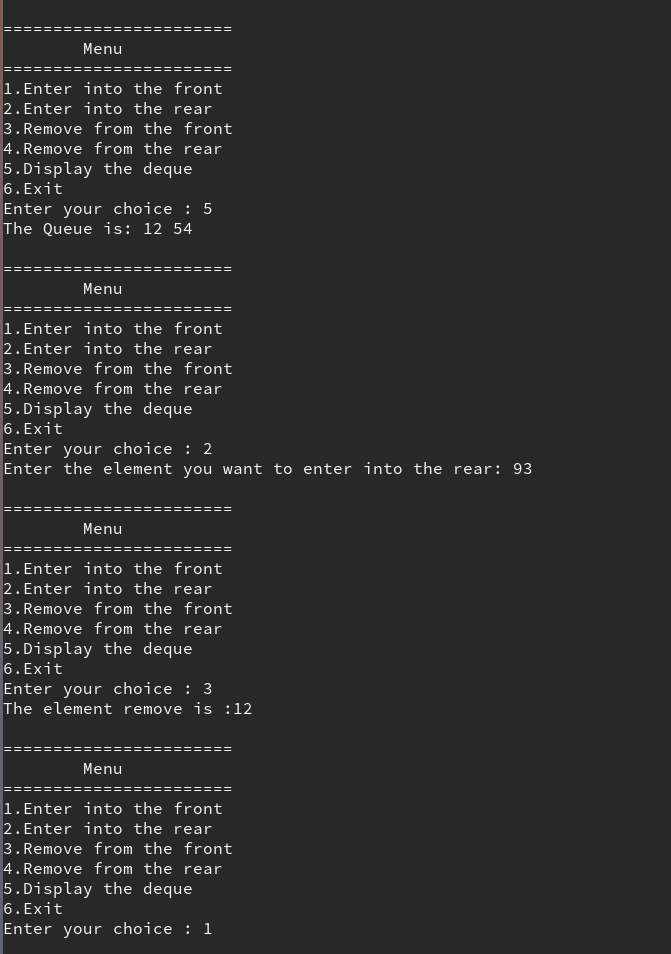
delQueue(myDeque);

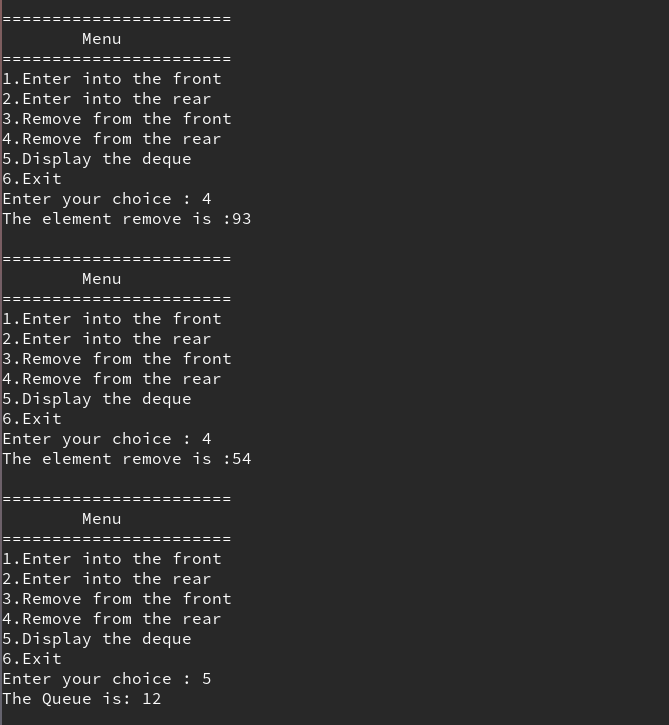
printf("\nExiting.....\n");

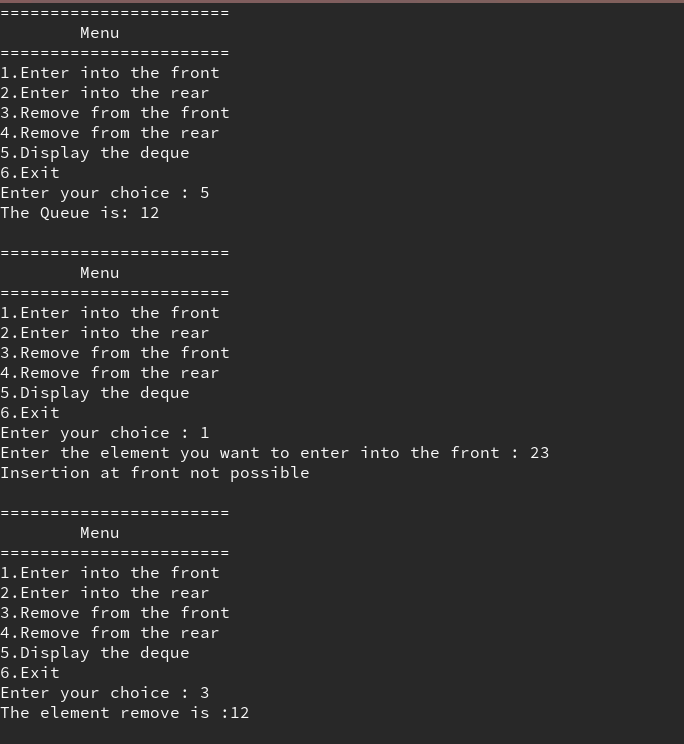
}

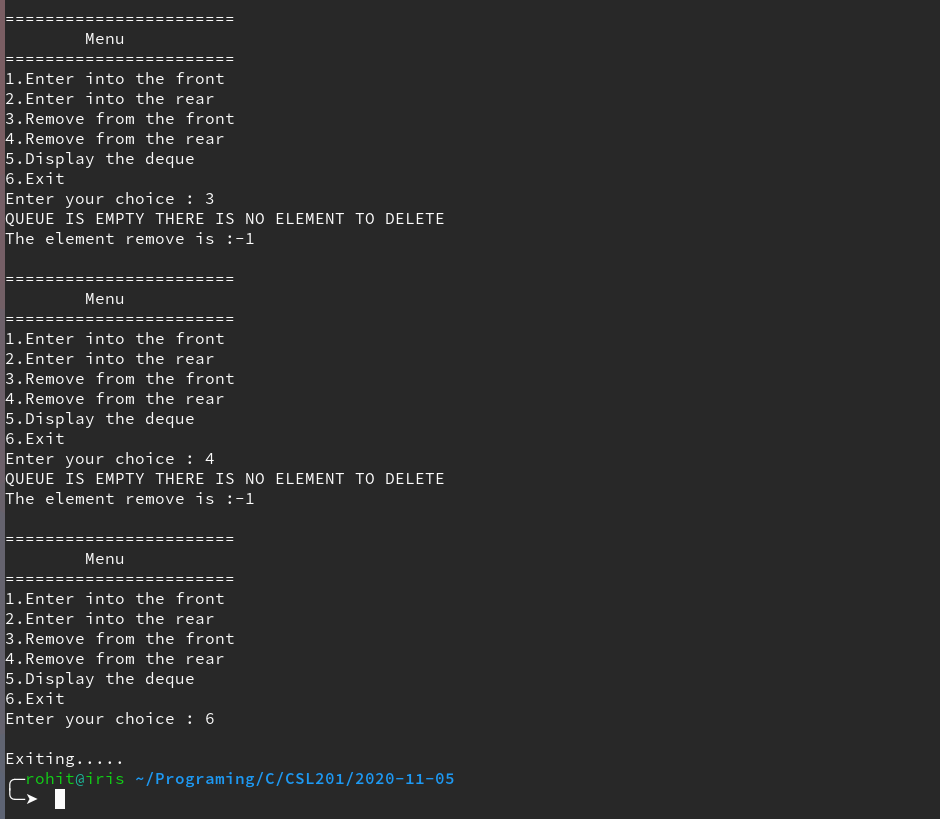
**Sample input and output:**

****

****

****

****

****

**Result:** the Program compiled successfully and the desired output was obtained.