1. Array Implementation of Priority Queue

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <stdbool.h>

#define MAX 6

int intArray[MAX];

int itemCount = 0;

int peek(){

return intArray[itemCount - 1];

}

bool isEmpty(){

return itemCount == 0;

}

bool isFull(){

return itemCount == MAX;

}

int size(){

return itemCount;

}

void insert(int data)

{

int i = 0;

if(!isFull()){

// if queue is empty, insert the data

if(itemCount == 0){

intArray[itemCount++] = data;

}else{

// start from the right end of the queue

for(i = itemCount - 1; i >= 0; i-- ){

// if data is larger, shift existing item to right end

if(data > intArray[i]){

intArray[i+1] = intArray[i];

}else{

break;

}

}

// insert the data

intArray[i+1] = data;

itemCount++;

}

}

}

int removeData(){

return intArray[--itemCount];

}

int main() {

/\* insert 5 items \*/

insert(3);

insert(5);

insert(9);

insert(1);

insert(12);

// ------------------

// index : 0 1 2 3 4

// ------------------

// queue : 12 9 5 3 1

insert(15);

// ---------------------

// index : 0 1 2 3 4 5

// ---------------------

// queue : 15 12 9 5 3 1

if(isFull()){

printf("Queue is full!\n");

}

// remove one item

int num = removeData();

printf("Element removed: %d\n",num);

// ---------------------

// index : 0 1 2 3 4

// ---------------------

// queue : 15 12 9 5 3

// insert more items

insert(16);

// ----------------------

// index : 0 1 2 3 4 5

// ----------------------

// queue : 16 15 12 9 5 3

// As queue is full, elements will not be inserted.

insert(17);

insert(18);

// ----------------------

// index : 0 1 2 3 4 5

// ----------------------

// queue : 16 15 12 9 5 3

printf("Element at front: %d\n",peek());

printf("----------------------\n");

printf("index : 5 4 3 2 1 0\n");

printf("----------------------\n");

printf("Queue: ");

while(!isEmpty()){

int n = removeData();

printf("%d ",n);

}

}

OUTPUT:

Queue is full!

Element removed: 1

Element at front: 3

----------------------

index : 5 4 3 2 1 0

----------------------

Queue: 3 5 9 12 15 16

2. Array Implementation of Deque

#define size 5

#include <stdio.h>

int deque[size];

int f=-1, r=-1;

// enqueue\_front function will insert thevalue from the front

void enqueue\_front(int x)

{

if((f==0 && r==size-1) || (f==r+1))

{

printf("deque is full");

}

else if((f==-1) && (r==-1))

{

f=r=0;

deque[f]=x;

}

else if(f==0)

{

f=size-1;

deque[f]=x;

}

else

{

f=f-1;

deque[f]=x;

}

}

// enqueue\_rear function will insert the valuefrom the rear

void enqueue\_rear(int x)

{

if((f==0 && r==size-1) || (f==r+1))

{

printf("deque is full");

}

else if((f==-1) && (r==-1))

{

r=0;

deque[r]=x;

}

else if(r==size-1)

{

r=0;

deque[r]=x;

}

else

{

r++;

deque[r]=x;

}

}

// display function prints all the value of deque.

void display()

{

int i=f;

printf("\n Elements in a deque : ");

while(i!=r)

{

printf("%d ",deque[i]);

i=(i+1)%size;

}

printf("%d",deque[r]);

}

// getfront function retrieves the first value of the deque.

void getfront()

{

if((f==-1) && (r==-1))

{

printf("Deque is empty");

}

else

{

printf("\nThe value of the front is: %d", deque[f]);

}

}

// getrear function retrieves the last value of the deque.

void getrear()

{

if((f==-1) && (r==-1))

{

printf("Deque is empty");

}

else

{

printf("\nThe value of the rear is: %d", deque[r]);

}

}

// dequeue\_front() function deletes the element from thefront

void dequeue\_front()

{

if((f==-1) && (r==-1))

{

printf("Deque is empty");

}

else if(f==r)

{

printf("\nThe deleted element is %d",

deque[f]);

f=-1;

r=-1;

}

else if(f==(size-1))

{

printf("\nThe deleted element is %d", deque[f]);

f=0;

}

else

{

printf("\nThe deleted element is %d", deque[f]);

f=f+1;

}

}

// dequeue\_rear() function deletes the element from the rear

void dequeue\_rear()

{

if((f==-1) && (r==-1))

{

printf("Deque is empty");

}

else if(f==r)

{

printf("\nThe deleted element is %d",

deque[r]);

f=-1;

r=-1;

}

else if(r==0)

{

printf("\nThe deleted element is %d", deque[r]);

r=size-1;

}

else

{

printf("\nThe deleted element is %d", deque[r]);

r=r-1;

}

}

int main()

{

// inserting a value from the front.

enqueue\_front(2);

// inserting a value from the front.

enqueue\_front(1);

// inserting a value from the rear.

enqueue\_rear(3);

// inserting a value from the rear.

enqueue\_rear(5);

// inserting a value from the rear.

enqueue\_rear(8);

// Calling the display function to retrieve the values of deque

display();

// Retrieve the front value

getfront();

// Retrieve the rear value.

getrear();

// deleting a value from the front

dequeue\_front();

//deleting a value from the rear

dequeue\_rear();

// Calling the display function to retrieve the values of deque

display();

return 0;

}

OUTPUT:

Elements in a deque : 1 2 3 5 8

The value of the front is: 1

The value of the rear is: 8

The deleted element is 1

The deleted element is 8

Elements in a deque : 2 3 5