*/****/Assignment-9 Data Structures Lab***

***//Create and perform different operations on Double-ended Queues using Linked List implementation***

#include <bits/stdc++.h>

#include<conio.h>

using namespace std;

// Node of a doubly linked list

struct Node

{

int data;

Node \*prev, \*next;

// Function to get a new node

static Node\* getnode(int data)

{

Node\* newNode = (Node\*)malloc(sizeof(Node));

newNode->data = data;

newNode->prev = newNode->next = NULL;

return newNode;

}

};

// A structure to represent a deque

class Deque

{

Node\* front;

Node\* rear;

int Size;

public:

Deque()

{

front = rear = NULL;

Size = 0;

}

// Operations on Deque

void insertFront(int data);

void insertRear(int data);

void deleteFront();

void deleteRear();

int getFront();

int getRear();

int size();

bool isEmpty();

void erase();

};

// Function to check whether deque

// is empty or not

bool Deque::isEmpty()

{

return (front == NULL);

}

// Function to return the number of

// elements in the deque

int Deque::size()

{

return Size;

}

// Function to insert an element

// at the front end

void Deque::insertFront(int data)

{

Node\* newNode = Node::getnode(data);

// If true then new element cannot be added

// and it is an 'Overflow' condition

if (newNode == NULL)

cout << "OverFlow\n";

else

{

// If deque is empty

if (front == NULL)

rear = front = newNode;

// Inserts node at the front end

else

{

newNode->next = front;

front->prev = newNode;

front = newNode;

}

// Increments count of elements by 1

Size++;

}

}

// Function to insert an element

// at the rear end

void Deque::insertRear(int data)

{

Node\* newNode = Node::getnode(data);

// If true then new element cannot be added

// and it is an 'Overflow' condition

if (newNode == NULL)

cout << "OverFlow\n";

else

{

// If deque is empty

if (rear == NULL)

front = rear = newNode;

// Inserts node at the rear end

else

{

newNode->prev = rear;

rear->next = newNode;

rear = newNode;

}

Size++;

}

}

// Function to delete the element

// from the front end

void Deque::deleteFront()

{

// If deque is empty then

// 'Underflow' condition

if (isEmpty())

cout << "UnderFlow\n";

// Deletes the node from the front end and makes

// the adjustment in the links

else

{

Node\* temp = front;

front = front->next;

// If only one element was present

if (front == NULL)

rear = NULL;

else

front->prev = NULL;

free(temp);

// Decrements count of elements by 1

Size--;

}

}

// Function to delete the element

// from the rear end

void Deque::deleteRear()

{

// If deque is empty then

// 'Underflow' condition

if (isEmpty())

cout << "UnderFlow\n";

// Deletes the node from the rear end and makes

// the adjustment in the links

else

{

Node\* temp = rear;

rear = rear->prev;

// If only one element was present

if (rear == NULL)

front = NULL;

else

rear->next = NULL;

free(temp);

// Decrements count of elements by 1

Size--;

}

}

// Function to return the element

// at the front end

int Deque::getFront()

{

// If deque is empty, then returns

// garbage value

if (isEmpty())

return -1;

return front->data;

}

// Function to return the element

// at the rear end

int Deque::getRear()

{

// If deque is empty, then returns

// garbage value

if (isEmpty())

return -1;

return rear->data;

}

// Function to delete all the elements

// from Deque

void Deque::erase()

{

rear = NULL;

while (front != NULL)

{

Node\* temp = front;

front = front->next;

free(temp);

}

Size = 0;

}

// Driver program to test above

int main()

{

Deque dq;

cout << "Insert element '5' at rear end\n";

dq.insertRear(5);

cout << "Insert element '10' at rear end\n";

dq.insertRear(10);

cout << "Rear end element: "

<< dq.getRear() << endl;

dq.deleteRear();

cout << "After deleting rear element new rear"

<< " is: " << dq.getRear() << endl;

cout << "Inserting element '15' at front end \n";

dq.insertFront(15);

cout << "Front end element: "

<< dq.getFront() << endl;

cout << "Number of elements in Deque: "

<< dq.size() << endl;

dq.deleteFront();

cout << "After deleting front element new "

<< "front is: " << dq.getFront() << endl;

getch();

return 0;

}

**Output:**

Insert element '5' at rear end

Insert element '10' at rear end

Rear end element: 10

After deleting rear element new rear is: 5

Inserting element '15' at front end

Front end element: 15

Number of elements in Deque: 2

After deleting front element new front is: 5