Important Introduction

Course Group on Telegram

https://t.me/+-JXnq208EZYwOTI0

Important Before You Start...

Important Before You Start...

Before you start with this course you should review the following from the previous courses:

-Linked List Operations

-Template Functions

-Template Classes

Requirements

هنعمل double linked list class وضروري تلتزم بال instructions والمسميات اللي بيديهالك

محتاج تعمل كلاس اسمه كده بالضبط MydblLinkedList

ال linked list بتكون doubly

```
clsDblLinkedList <int> MydblLinkedList;
```

وبتروح لل header files وبتعمل filter جديد اسمه DS وبتحط جواه الكلاس اللي فوق ده

الكلاس بنخليه يدعم ال template class

الكلاس ده بيكون فيه ال function اللي اسمها insert at the beginning

```
MydblLinkedList.InsertAtBeginning(5);
```

و function تانيه print list

```
MydblLinkedList.PrintList();
```

وال find

```
clsDblLinkedList<int>::Node* N1 = MydblLinkedList.Find(2);
```

وال insert after

```
MydblLinkedList.InsertAfter(N1, 500);
```

وال insert at end

```
MydblLinkedList.InsertAtEnd(700);
```

delete nodeو

```
MydblLinkedList.DeleteNode(N2);
```

delete first node وال

```
MydblLinkedList.DeleteFirstNode();
```

delete last node وال

```
MydblLinkedList.DeleteLastNode();
```

```
int main()
{

    clsDblLinkedList <int> MydblLinkedList;

    MydblLinkedList.InsertAtBeginning(5);
    MydblLinkedList.InsertAtBeginning(4);
    MydblLinkedList.InsertAtBeginning(3);
    MydblLinkedList.InsertAtBeginning(2);
    MydblLinkedList.InsertAtBeginning(1);

    cout << "\nLinked List Contenet:\n";
    MydblLinkedList.PrintList();

    clsDblLinkedList<int>::Node* N1 = MydblLinkedList.Find(2);

    if (N1 != NULL)
        cout << "\nNode with value 2 is Found :-)\n";
    else
        cout << "\nNode Is not found :-(\n";

    MydblLinkedList.InsertAfter(N1, 500);
    cout << "\nAfter Inserting 500 after 2:\n";
    MydblLinkedList.PrintList();

    MydblLinkedList.InsertAtEnd(700);
    cout << "\nAfter Inserting 700 at end:\n";
```

```
clsDblLinkedList<int>::Node* N2 = MydblLinkedList.Find(4);
MydblLinkedList.DeleteNode(N2);
cout << "\nAfter Deleting 4:\n";
MydblLinkedList.PrintList();

MydblLinkedList.DeleteFirstNode();
cout << "\nAfter Deleting First Node:\n";
MydblLinkedList.PrintList();

cout << "\nAfter Deleting Last Node:\n";
MydblLinkedList.DeleteLastNode();
MydblLinkedList.PrintList();
```

```
C:\Users\USER\source\repos\Main2\x64\Debug\Main2.exe

Linked List Contenet:
1 2 3 4 5

Node with value 2 is Found :-)

After Inserting 500 after 2:
1 2 500 3 4 5

After Inserting 700 at end:
1 2 500 3 4 5 700

After Deleting 4:
1 2 500 3 5 700

After Deleting First Node:
2 500 3 5 700

After Deleting Last Node:
2 500 3 5
```

Solution

بيديك hint صغير كده وهوا انك خلي ال head بتاعك ثابت ودايما واقف عند اول node ولما تحب تتحرك ابقي اعمل مؤشر قيمته بتساوي ال head يعني خد نسخه من ال head واشتغل عليها

ده كلاس ال linked list

```
//ProgrammingAdvices.com
```

```
//Mohammed Abu-Hadhoud

#pragma once
#include <iostream>
using namespace std;


template <class T>
class clsDblLinkedList
{

public:

    class Node
    {

    public:
        T value;
        Node* next;
        Node* prev;
    };

    Node* head = NULL;

    void InsertAtBeginning(T value)
    {

        /*
            1-Create a new node with the desired value.
            2-Set the next pointer of the new node to the current head of the list.
            3-Set the previous pointer of the current head to the new node.
            4-Set the new node as the new head of the list.
        */

        Node* newNode = new Node();
        newNode->value = value;
        newNode->next = head;
        newNode->prev = NULL;

        if (head != NULL) {
            head->prev = newNode;
        }
        head = newNode;


    }

    // Print the linked list
    void PrintList()

    {
        Node* Current = head;

        while (Current != NULL) {
            cout << Current->value << " ";
            Current = Current->next;
        }
        cout << "\n";


    }

    Node* Find(T Value)
    {
        Node* Current = head;
        while (Current != NULL) {

            if (Current->value == Value)
                return Current;

            Current = Current->next;
```

```
        }

        return NULL;

    }

    void InsertAfter(Node* current, T value) {


        /*  1 - Create a new node with the desired value.
             2-Set the next pointer of the new node to the next node of the current node.
             3-Set the previous pointer of the new node to the current node.
             4-Set the next pointer of the current node to the new node.
             5-Set the previous pointer of the next node to the new node(if it exists).
        */

        Node* newNode = new Node();
        newNode->value = value;
        newNode->next = current->next;
        newNode->prev = current;

        if (current->next != NULL) {
            current->next->prev = newNode;
        }
        current->next = newNode;


    }

    void InsertAtEnd(T value) {

        /*
            1-Create a new node with the desired value.
            2-Traverse the list to find the last node.
            3-Set the next pointer of the last node to the new node.
            4-Set the previous pointer of the new node to the last node.
        */


        Node* newNode = new Node();
        newNode->value = value;
        newNode->next = NULL;
        if (head == NULL) {
            newNode->prev = NULL;
            head = newNode;
        }
        else {
            Node* current = head;
            while (current->next != NULL) {
                current = current->next;
            }
            current->next = newNode;
            newNode->prev = current;
        }


    }

    void DeleteNode(Node*& NodeToDelete) {

        /*
            1-Set the next pointer of the previous node to the next pointer of the current
node.
            2-Set the previous pointer of the next node to the previous pointer of the
current node.
            3-Delete the current node.
        */
        if (head == NULL || NodeToDelete == NULL) {
            return;
        }
        if (head == NodeToDelete) {
```

```
            head = NodeToDelete->next;
        }
        if (NodeToDelete->next != NULL) {
            NodeToDelete->next->prev = NodeToDelete->prev;
        }
        if (NodeToDelete->prev != NULL) {
            NodeToDelete->prev->next = NodeToDelete->next;
        }
        delete NodeToDelete;


    }

    void DeleteFirstNode()
    {

        /*
            1-Store a reference to the head node in a temporary variable.
            2-Update the head pointer to point to the next node in the list.
            3-Set the previous pointer of the new head to NULL.
            4-Delete the temporary reference to the old head node.
        */

        if (head == NULL) {
            return;
        }
        Node* temp = head;
        head = head->next;
        if (head != NULL) {
            head->prev = NULL;
        }
        delete temp;

    }

    void DeleteLastNode() {

        /*
            1-Traverse the list to find the last node.
            2-Set the next pointer of the second-to-last node to NULL.
            3-Delete the last node.
        */

        if (head == NULL) {
            return;
        }

        if (head->next == NULL) {
            delete head;
            head = NULL;
            return;
        }

        Node* current = head;
        // we need to find the node before last node.
        while (current->next->next != NULL)
        {
            current = current->next;
        }

        Node* temp = current->next;
        current->next = NULL;
        delete temp;

    }

};
```

وده ال main

```
//ProgrammingAdvices.com
```

```
//Mohammed Abu-Hadhoud

#include <iostream>
#include "clsDblLinkedList.h"

using namespace std;


int main()
{

    clsDblLinkedList <int> MydblLinkedList;

    MydblLinkedList.InsertAtBeginning(5);
    MydblLinkedList.InsertAtBeginning(4);
    MydblLinkedList.InsertAtBeginning(3);
    MydblLinkedList.InsertAtBeginning(2);
    MydblLinkedList.InsertAtBeginning(1);

    cout << "\nLinked List Contenet:\n";
    MydblLinkedList.PrintList();

    clsDblLinkedList<int>::Node* N1 = MydblLinkedList.Find(2);

    if (N1 != NULL)
        cout << "\nNode with value 2 is Found :-)\n";
    else
        cout << "\nNode Is not found :-(\n";

    MydblLinkedList.InsertAfter(N1, 500);
    cout << "\nAfter Inserting 500 after 2:\n";
    MydblLinkedList.PrintList();

    MydblLinkedList.InsertAtEnd(700);
    cout << "\nAfter Inserting 700 at end:\n";
    MydblLinkedList.PrintList();

    clsDblLinkedList<int>::Node* N2 = MydblLinkedList.Find(4);
    MydblLinkedList.DeleteNode(N2);
    cout << "\nAfter Deleting 4:\n";
    MydblLinkedList.PrintList();

    MydblLinkedList.DeleteFirstNode();
    cout << "\nAfter Deleting First Node:\n";
    MydblLinkedList.PrintList();

    cout << "\nAfter Deleting Last Node:\n";
    MydblLinkedList.DeleteLastNode();
    MydblLinkedList.PrintList();

    system("pause>0");

}
```

Extension 1 - Requirements

عاوزين نعمل function تجيبلك ال size بتاع ال list بتجيب عدد العناصر بتاعت ال list بس ماتلفش عالعناصر كلها عشان تجيب العدد

```
int main()
{

    clsDblLinkedList <int> MydblLinkedList;

    MydblLinkedList.InsertAtBeginning(5);
    MydblLinkedList.InsertAtBeginning(4);
    MydblLinkedList.InsertAtBeginning(3);
    MydblLinkedList.InsertAtBeginning(2);
    MydblLinkedList.InsertAtBeginning(1);

    cout << "\nLinked List Contenet:\n";
    MydblLinkedList.PrintList();
```

```
cout << "\nNumber of items in the linked list = " << MydblLinkedList.Size();
```

```
Linked List Contenet:
1 2 3 4 5

Number of items in the linked list = 5
```

Extension 1 - Solution

لو قعدت تلف علي كل العناصر وتعدهم كده ال big o بتاعك هوا o(n) وده مش كويس

فيه حل احسن وده ال big o بتاعه هوا o(1)

فكرته باختصار هوا انك تعرف متغير من النوع int ومع كل عمليه اضافه او حذف بتزود او بتنقص الرقم اللي موجود في المتغير

وده كود الكلاس

```
#pragma once

#include <iostream>
using namespace std;


template <class T>
class clsDblLinkedList
{

protected:
    int _Size = 0;
```

```
public:

    class Node
    {

    public:
        T value;
        Node* next;
        Node* prev;
    };

    Node* head = NULL;

    void InsertAtBeginning(T value)
    {

        /*
            1-Create a new node with the desired value.
            2-Set the next pointer of the new node to the current head of the list.
            3-Set the previous pointer of the current head to the new node.
            4-Set the new node as the new head of the list.
        */

        Node* newNode = new Node();
        newNode->value = value;
        newNode->next = head;
        newNode->prev = NULL;

        if (head != NULL) {
            head->prev = newNode;
        }
        head = newNode;
        _Size++;

    }

    // Print the linked list
    void PrintList()

    {
        Node* Current = head;

        while (Current != NULL) {
            cout << Current->value << " ";
            Current = Current->next;
        }
        cout << "\n";
        delete Current;

    }

    Node* Find(T Value)
    {
        Node* Current = head;
        while (Current != NULL) {

            if (Current->value == Value)
                return Current;

            Current = Current->next;
        }

        return NULL;

    }

    void InsertAfter(Node* current, T value) {


        /*  1 - Create a new node with the desired value.
             2-Set the next pointer of the new node to the next node of the current node.
```

```
            3-Set the previous pointer of the new node to the current node.
            4-Set the next pointer of the current node to the new node.
            5-Set the previous pointer of the next node to the new node(if it exists).
        */

        Node* newNode = new Node();
        newNode->value = value;
        newNode->next = current->next;
        newNode->prev = current;

        if (current->next != NULL) {
            current->next->prev = newNode;
        }
        current->next = newNode;
        _Size++;

    }

    void InsertAtEnd(T value) {

        /*
            1-Create a new node with the desired value.
            2-Traverse the list to find the last node.
            3-Set the next pointer of the last node to the new node.
            4-Set the previous pointer of the new node to the last node.
        */


        Node* newNode = new Node();
        newNode->value = value;
        newNode->next = NULL;
        if (head == NULL) {
            newNode->prev = NULL;
            head = newNode;
        }
        else {
            Node* current = head;
            while (current->next != NULL) {
                current = current->next;
            }
            current->next = newNode;
            newNode->prev = current;
        }
        _Size++;

    }

    void DeleteNode(Node*& NodeToDelete) {

        /*
            1-Set the next pointer of the previous node to the next pointer of the current
node.
            2-Set the previous pointer of the next node to the previous pointer of the
current node.
            3-Delete the current node.
        */
        if (head == NULL || NodeToDelete == NULL) {
            return;
        }
        if (head == NodeToDelete) {
            head = NodeToDelete->next;
        }
        if (NodeToDelete->next != NULL) {
            NodeToDelete->next->prev = NodeToDelete->prev;
        }
        if (NodeToDelete->prev != NULL) {
            NodeToDelete->prev->next = NodeToDelete->next;
        }
        delete NodeToDelete;

        _Size--;
```

```
    }

    void DeleteFirstNode()
    {

        /*
            1-Store a reference to the head node in a temporary variable.
            2-Update the head pointer to point to the next node in the list.
            3-Set the previous pointer of the new head to NULL.
            4-Delete the temporary reference to the old head node.
        */

        if (head == NULL) {
            return;
        }
        Node* temp = head;
        head = head->next;
        if (head != NULL) {
            head->prev = NULL;
        }
        delete temp;
        _Size--;
    }

    void DeleteLastNode() {

        /*
            1-Traverse the list to find the last node.
            2-Set the next pointer of the second-to-last node to NULL.
            3-Delete the last node.
        */

        if (head == NULL) {
            return;
        }

        if (head->next == NULL) {
            delete head;
            head = NULL;
            return;
        }

        Node* current = head;
        // we need to find the node before last node.
        while (current->next->next != NULL)
        {
            current = current->next;
        }

        Node* temp = current->next;
        current->next = NULL;
        delete temp;
        _Size--;
    }

    int Size()
    {
        return _Size;
    }



};
```

وده كود ال main

```
//ProgrammingAdvices.com
//Mohammed Abu-Hadhoud

#include <iostream>
#include "clsDblLinkedList.h"
```

```
using namespace std;


int main()
{

    clsDblLinkedList <int> MydblLinkedList;

    MydblLinkedList.InsertAtBeginning(5);
    MydblLinkedList.InsertAtBeginning(4);
    MydblLinkedList.InsertAtBeginning(3);
    MydblLinkedList.InsertAtBeginning(2);
    MydblLinkedList.InsertAtBeginning(1);

    cout << "\nLinked List Contenet:\n";
    MydblLinkedList.PrintList();

    cout << "\nNumber of items in the linked list = " << MydblLinkedList.Size();

    system("pause>0");

}
```

Extension 2 - Requirements

عاوزين نعمل method اسمها is empty

```
if (MydblLinkedList.IsEmpty())
    cout << "\n\nYes List is Empty.\n";
else
    cout << "\n\nNo List is not Empty.\n";
```

Extension 2 - Solution

ده التعديل علي كود الكلاس

```
    bool IsEmpty()
    {
        return (_Size == 0 ? true : false);

    }
```

ده كود ال main

```
//ProgrammingAdvices.com
//Mohammed Abu-Hadhoud

#include <iostream>
#include "clsDblLinkedList.h"

using namespace std;


int main()
{

    clsDblLinkedList <int> MydblLinkedList;
```

```
    if (MydblLinkedList.IsEmpty())
        cout << "\n\nYes List is Empty.\n";
    else
        cout << "\n\nNo List is not Empty.\n";

    MydblLinkedList.InsertAtBeginning(5);
    MydblLinkedList.InsertAtBeginning(4);
    MydblLinkedList.InsertAtBeginning(3);
    MydblLinkedList.InsertAtBeginning(2);
    MydblLinkedList.InsertAtBeginning(1);

    cout << "\nLinked List Contenet:\n";
    MydblLinkedList.PrintList();

    cout << "\nNumber of items in the linked list = " << MydblLinkedList.Size();

    if (MydblLinkedList.IsEmpty())
        cout << "\n\nYes List is Empty.\n";
    else
        cout << "\n\nNo List is not Empty.\n";


    system("pause>0");

}
```

Extension 3 - Requirements

عاوزين نعمل function تمسح كل العناصر في ال linked list

```
MydblLinkedList.Clear();
```

```
int main()
{

    clsDblLinkedList <int> MydblLinkedList;

    MydblLinkedList.InsertAtBeginning(5);
    MydblLinkedList.InsertAtBeginning(4);
    MydblLinkedList.InsertAtBeginning(3);
    MydblLinkedList.InsertAtBeginning(2);
    MydblLinkedList.InsertAtBeginning(1);

    cout << "\nLinked List Contenet:\n";
    MydblLinkedList.PrintList();
```

```
cout << "\nLinked List Contenet:\n";
MydblLinkedList.PrintList();

cout << "\nNumber of items in the linked list = " << MydblLinkedList.Size();

cout << "\nExecuting .Clear()";
MydblLinkedList.Clear();
cout << "\nNumber of items in the linked list = " << MydblLinkedList.Size();
```

```
Linked List Contenet:
1 2 3 4 5

Number of items in the linked list = 5
Executing .Clear()
Number of items in the linked list = 0
```

Extension 3 - Solution

ده التغيير عالكلاس

```
  void Clear()
  {
      while (_Size > 0)
      {
          DeleteFirstNode();
      }
  }
```

وده ال main

```
//ProgrammingAdvices.com
//Mohammed Abu-Hadhoud

#include <iostream>
#include "clsDblLinkedList.h"

using namespace std;


int main()
{

    clsDblLinkedList <int> MydblLinkedList;

    MydblLinkedList.InsertAtBeginning(5);
    MydblLinkedList.InsertAtBeginning(4);
    MydblLinkedList.InsertAtBeginning(3);
    MydblLinkedList.InsertAtBeginning(2);
    MydblLinkedList.InsertAtBeginning(1);

    cout << "\nLinked List Contenet:\n";
    MydblLinkedList.PrintList();

    cout << "\nNumber of items in the linked list = " << MydblLinkedList.Size();

    cout << "\nExecuting .Clear()";
    MydblLinkedList.Clear();
    cout << "\nNumber of items in the linked list = " << MydblLinkedList.Size();


    system("pause>0");

}
```

Extension 4 - Requirements

عاوزين نعمل reverse للعناصر اللي في الأول ييجي في الاخر واللي في الاخر ييجي في الأول

وبعدين نطبعهم

```
clsDblLinkedList <int> MydblLinkedList;

MydblLinkedList.InsertAtBeginning(5);
MydblLinkedList.InsertAtBeginning(4);
MydblLinkedList.InsertAtBeginning(3);
MydblLinkedList.InsertAtBeginning(2);
MydblLinkedList.InsertAtBeginning(1);

cout << "\nLinked List Contenet:\n";
MydblLinkedList.PrintList();

MydblLinkedList.Reverse();
cout << "\nLinked List Contenet after reverse:\n";
MydblLinkedList.PrintList();
```

```
Linked List Contenet:
1 2 3 4 5

Linked List Contenet after reverse:
5 4 3 2 1
```

Extension 4 - Solution

الفكره كلها انك بتمسك كل node وتبدل ال next بال prev

شوف الجدول ده وانت تفهم

value	1	2	3	4	5
next	2	3	4	5	0
prev	0	1	2	3	4
swap					
value	5	4	3	2	1
next	4	3	2	1	0
prev	0	5	4	3	2

ده التعديل علي كود الكلاس

```
void  Reverse()
{
    Node* current = head;
    Node* temp = nullptr;
    while (current != nullptr) {
        temp = current->prev;
        current->prev = current->next;
        current->next = temp;
        current = current->prev;
    }

    if (temp != nullptr) {
        head = temp->prev;
    }
}
```

وده ال main

```
//ProgrammingAdvices.com
//Mohammed Abu-Hadhoud

#include <iostream>
#include "clsDblLinkedList.h"

using namespace std;


int main()
{

    clsDblLinkedList <int> MydblLinkedList;

    MydblLinkedList.InsertAtBeginning(5);
    MydblLinkedList.InsertAtBeginning(4);
    MydblLinkedList.InsertAtBeginning(3);
    MydblLinkedList.InsertAtBeginning(2);
    MydblLinkedList.InsertAtBeginning(1);

    cout << "\nLinked List Contenet:\n";
    MydblLinkedList.PrintList();

    MydblLinkedList.Reverse();

    cout << "\nLinked List Contenet after reverse:\n";
    MydblLinkedList.PrintList();
```

```
    system("pause>0");

}
```

Extension 5 - Requirements

عاوزين ندور عال node بال index يعني اقولك getnode(2) تجيبلي ال node بال next بال prev

```
clsDblLinkedList <int> MydblLinkedList;

MydblLinkedList.InsertAtBeginning(5);
MydblLinkedList.InsertAtBeginning(4);
MydblLinkedList.InsertAtBeginning(3);
MydblLinkedList.InsertAtBeginning(2);
MydblLinkedList.InsertAtBeginning(1);

cout << "\nLinked List Contenet:\n";
MydblLinkedList.PrintList();


clsDblLinkedList <int> ::Node *N;

N = MydblLinkedList.GetNode(2);

cout << "\nNode Value is: " << N->value ;
```

يعني تستدعي ال function وتديها الرقم 2 تجيبلك ال node الثالثه لانها بتعد من الصفر

Extension 5 - Solution

بالنسبالي انا عاملها ب for loop

ده التعديل علي الكود

```
Node* GetNode(int Index)
{

    int Counter = 0;

    if (Index > _Size - 1 || Index < 0)
        return NULL;

    Node* Current = head;
    while (Current != NULL && (Current->next != NULL)) {

        if (Counter == Index)
            break;
```

```
            Current = Current->next;
            Counter++;

        }

        return Current;
    }
```

وده ال main

```
//ProgrammingAdvices.com
//Mohammed Abu-Hadhoud

#include <iostream>
#include "clsDblLinkedList.h"

using namespace std;


int main()
{

    clsDblLinkedList <int> MydblLinkedList;

    MydblLinkedList.InsertAtBeginning(5);
    MydblLinkedList.InsertAtBeginning(4);
    MydblLinkedList.InsertAtBeginning(3);
    MydblLinkedList.InsertAtBeginning(2);
    MydblLinkedList.InsertAtBeginning(1);

    cout << "\nLinked List Contenet:\n";
    MydblLinkedList.PrintList();


    clsDblLinkedList <int> ::Node* N;

    N = MydblLinkedList.GetNode(1);

    cout << "\nNode Value is: " << N->value;



    system("pause>0");

}
```

Extension 6 - Requirements

عاوزين نعمل get item ترجعلنا ال value نفسها مش ال node

```
clsDblLinkedList <int> MydblLinkedList;

MydblLinkedList.InsertAtBeginning(5);
MydblLinkedList.InsertAtBeginning(4);
MydblLinkedList.InsertAtBeginning(3);
MydblLinkedList.InsertAtBeginning(2);
MydblLinkedList.InsertAtBeginning(1);

cout << "\nLinked List Contenet:\n";
MydblLinkedList.PrintList();


cout << "\nItem(2) Value is: " << MydblLinkedList.GetItem(2);
```

```
Linked List Contenet:
1 2 3 4 5

Item(2) Value is: 3
```

Extension 6 - Solution

ده التعديل علي الكلاس

```
T GetItem(int Index)
{

    Node* ItemNode = GetNode(Index);

    if (ItemNode == NULL)
        return NULL;
    else
        return ItemNode->value;

}
```

وده ال main

```
//ProgrammingAdvices.com
//Mohammed Abu-Hadhoud

#include <iostream>
#include "clsDblLinkedList.h"

using namespace std;


int main()
{

    clsDblLinkedList <int> MydblLinkedList;

    MydblLinkedList.InsertAtBeginning(5);
    MydblLinkedList.InsertAtBeginning(4);
    MydblLinkedList.InsertAtBeginning(3);
    MydblLinkedList.InsertAtBeginning(2);
    MydblLinkedList.InsertAtBeginning(1);

    cout << "\nLinked List Contenet:\n";
    MydblLinkedList.PrintList();
```

```
    cout << "\nItem(2) Value is: " << MydblLinkedList.GetItem(2);

    system("pause>0");

}
```

Extension 7 - Requirements

عاوزين نعمل حاجه تعدل علي ال value بتاع ال item عن طريق ال index بتاعه

```
clsDblLinkedList <int> MydblLinkedList;

MydblLinkedList.InsertAtBeginning(5);
MydblLinkedList.InsertAtBeginning(4);
MydblLinkedList.InsertAtBeginning(3);
MydblLinkedList.InsertAtBeginning(2);
MydblLinkedList.InsertAtBeginning(1);

cout << "\nLinked List Contenet:\n";
MydblLinkedList.PrintList();

MydblLinkedList.UpdateItem(2, 500);

cout << "\nAfter Updating Item(2): " <<"\n";
MydblLinkedList.PrintList();
```

```
Linked List Contenet:
1 2 3 4 5

After Updating Item(2):
1 2 500 4 5
```

Extension 7 - Solution

ده التعديل علي الكلاس

```
bool UpdateItem(int Index, T NewValue)
{
    Node* ItemNode = GetNode(Index);

    if (ItemNode != NULL)
    {
        ItemNode->value = NewValue;
        return true;
    }
    else
```

```
        return false;

    }
```

وده ال main

```
//ProgrammingAdvices.com
//Mohammed Abu-Hadhoud

#include <iostream>
#include "clsDblLinkedList.h"

using namespace std;


int main()
{

    clsDblLinkedList <int> MydblLinkedList;

    MydblLinkedList.InsertAtBeginning(5);
    MydblLinkedList.InsertAtBeginning(4);
    MydblLinkedList.InsertAtBeginning(3);
    MydblLinkedList.InsertAtBeginning(2);
    MydblLinkedList.InsertAtBeginning(1);

    cout << "\nLinked List Contenet:\n";
    MydblLinkedList.PrintList();

    MydblLinkedList.UpdateItem(2, 500);

    cout << "\nAfter Updating Item(2): " << "\n";
    MydblLinkedList.PrintList();



    system("pause>0");

}
```

Extension 8 - Requirements

عاوزين نعمل overloading لل insert after بحيث انها تاخد index بدل ماتاخد node

```
clsDblLinkedList <int> MydblLinkedList;

MydblLinkedList.InsertAtBeginning(5);
MydblLinkedList.InsertAtBeginning(4);
MydblLinkedList.InsertAtBeginning(3);
MydblLinkedList.InsertAtBeginning(2);
MydblLinkedList.InsertAtBeginning(1);

cout << "\nLinked List Contenet:\n";
MydblLinkedList.PrintList();

MydblLinkedList.InsertAfter(1, 500);

cout << "\nAfter Insert: " <<"\n";
MydblLinkedList.PrintList();
```

```
Linked List Contenet:
1 2 3 4 5

After Insert:
1 2 500 3 4 5
```

Extension 8 - Solution

ده التعديل علي الكلاس

```
    bool InsertAfter(int Index, T value) {

        Node* ItemNode = GetNode(Index);

        if (ItemNode != NULL)
        {
            InsertAfter(ItemNode,value);
            return true;
        }
        else
            return false;
    }
```

وده ال main

```
//ProgrammingAdvices.com
//Mohammed Abu - Hadhoud

#include <iostream>
#include "clsDblLinkedList.h"
```

```
using namespace std;


int main()
{

    clsDblLinkedList <int> MydblLinkedList;

    MydblLinkedList.InsertAtBeginning(5);
    MydblLinkedList.InsertAtBeginning(4);
    MydblLinkedList.InsertAtBeginning(3);
    MydblLinkedList.InsertAtBeginning(2);
    MydblLinkedList.InsertAtBeginning(1);

    cout <<"\nLinked List Contenet\n";
    MydblLinkedList.PrintList();

    MydblLinkedList.InsertAfter(4, 500);

    cout << "\nAfter Insert\n";
    MydblLinkedList.PrintList();



    system("pause>0");

}
```

Project 2

Requirements

مش عاوزين نستخدم ال queue اللي موجود في المكتبه stl عاوزين نعمل ال queue بتاعنا

```
#include "clsMyQueue.h"

using namespace std;

int main()
{

    clsMyQueue <int> MyQueue;

    MyQueue.push(10);
    MyQueue.push(20);
    MyQueue.push(30);
    MyQueue.push(40);
    MyQueue.push(50);


    cout << "\nQueue: \n";
    MyQueue.Print();

    cout << "\nQueue Size: " << MyQueue.Size() ;
    cout << "\nQueue Front: " << MyQueue.front() ;
    cout << "\nQueue Back: " << MyQueue.back() ;
```

```
Queue:
10 20 30 40 50

Queue Size: 5
Queue Front: 10
Queue Back: 50

Queue after pop() :
20 30 40 50
```

```
MyQueue.pop();

cout << "\n\nQueue after pop() : \n";
MyQueue.Print();
```

Solution

الفكره اني هعمل object من الكلاس بتاع ال linked list واستدعي الشغل اللي فيه

وده اسمه composition

ده كود الكلاس بتاع ال queue

```
#pragma once

#include <iostream>
#include "clsDblLinkedList.h"


using namespace std;
template <class T>

class clsMyQueue
{

protected:
	clsDblLinkedList <T> _MyList;

public:

	void push(T Item)
	{
		_MyList.InsertAtEnd(Item);
	}


	void pop()
	{
		_MyList.DeleteFirstNode();
	}

	void Print()
	{
		_MyList.PrintList();
	}

	int Size()
	{
```

```
            return _MyList.Size();
        }

        bool IsEmpty()
        {
            return _MyList.IsEmpty();
        }

        T front()
        {
            return _MyList.GetItem(0);
        }

        T back()
        {
            return _MyList.GetItem(Size() - 1);
        }



};
```

وده ال main

```
//ProgrammingAdvices.com
//Mohammed Abu-Hadhoud

#include <iostream>
#include "clsMyQueue.h"

using namespace std;

int main()
{

    clsMyQueue <int> MyQueue;

    MyQueue.push(10);
    MyQueue.push(20);
    MyQueue.push(30);
    MyQueue.push(40);
    MyQueue.push(50);


    cout << "\nQueue: \n";
    MyQueue.Print();

    cout << "\nQueue Size: " << MyQueue.Size();
    cout << "\nQueue Front: " << MyQueue.front();
    cout << "\nQueue Back: " << MyQueue.back();

    MyQueue.pop();

    cout << "\n\nQueue after pop() : \n";
    MyQueue.Print();




    system("pause>0");

}
```

Extensions 1 to 7 Requirements

عاوزين نزود الشغل ده

```
//Extension #1
cout << "\n\n Item(2) : " << MyQueue.GetItem(2);

//Extension #2
MyQueue.Reverse();
cout << "\n\nQueue after reverse() : \n";
MyQueue.Print();

//Extension #3
MyQueue.UpdateItem(2, 600);
cout << "\n\nQueue after updating Item(2) to 600 : \n";
MyQueue.Print();
```

```
//Extension #4
MyQueue.InsertAfter(2, 800);
cout << "\n\nQueue after Inserting 800 after Item(2) : \n";
MyQueue.Print();


//Extension #5
MyQueue.InsertAtFront(1000);
cout << "\n\nQueue after Inserting 1000 at front: \n";
MyQueue.Print();

//Extension #6
MyQueue.InsertAtBack(2000);
cout << "\n\nQueue after Inserting 2000 at back: \n";
MyQueue.Print();

//Extension #7
MyQueue.Clear();
cout << "\n\nQueue after Clear(): \n";
MyQueue.Print();
```

```
Queue:
10 20 30 40 50

Queue Size: 5
Queue Front: 10
Queue Back: 50

Queue after pop() :
20 30 40 50


 Item(2) : 40

Queue after reverse() :
50 40 30 20


Queue after updating Item(2) to 600 :
50 40 600 20


Queue after Inserting 800 after Item(2) :
50 40 600 800 20


Queue after Inserting 1000 at front:
1000 50 40 600 800 20
```

```
Queue after Inserting 2000 at back:
1000 50 40 600 800 20 2000


Queue after Clear():
```

Extension 1 to 7 Solution

ده كود الكلاس

```
#pragma once

#include <iostream>
#include "clsDblLinkedList.h"


using namespace std;
template <class T>

class clsMyQueue
{

protected:
	clsDblLinkedList <T> _MyList;

public:

	void push(T Item)
	{
		_MyList.InsertAtEnd(Item);
	}
```

```
void pop()
{
        _MyList.DeleteFirstNode();
}

void Print()
{
        _MyList.PrintList();
}

int Size()
{
        return _MyList.Size();
}

bool IsEmpty()
{
        return _MyList.IsEmpty();
}

T front()
{
        return _MyList.GetItem(0);
}

T back()
{
        return _MyList.GetItem(Size() - 1);
}

T GetItem(int Index)
{
        return _MyList.GetItem(Index);

}

void Reverse()
{

        _MyList.Reverse();

}

void UpdateItem(int Index, T NewValue)
{
        _MyList.UpdateItem(Index, NewValue);

}

void InsertAfter(int Index, T NewValue)
{
        _MyList.InsertAfter(Index, NewValue);
}


void InsertAtFront(T Value)
{
        _MyList.InsertAtBeginning(Value);

}

void InsertAtBack(T Value)
{
        _MyList.InsertAtEnd(Value);


}


void Clear()
{
```

```
            _MyList.Clear();
        }



};
```

وده كود ال main

```
//ProgrammingAdvices.com
//Mohammed Abu-Hadhoud

#include <iostream>
#include "clsMyQueue.h"

using namespace std;

int main()
{

    clsMyQueue <int> MyQueue;

    MyQueue.push(10);
    MyQueue.push(20);
    MyQueue.push(30);
    MyQueue.push(40);
    MyQueue.push(50);


    cout << "\nQueue: \n";
    MyQueue.Print();

    cout << "\nQueue Size: " << MyQueue.Size();
    cout << "\nQueue Front: " << MyQueue.front();
    cout << "\nQueue Back: " << MyQueue.back();

    MyQueue.pop();

    cout << "\n\nQueue after pop() : \n";
    MyQueue.Print();

    //Extension #1
    cout << "\n\n Item(2) : " << MyQueue.GetItem(2);

    //Extension #2
    MyQueue.Reverse();
    cout << "\n\nQueue after reverse() : \n";
    MyQueue.Print();

    //Extension #3
    MyQueue.UpdateItem(2, 600);
    cout << "\n\nQueue after updating Item(2) to 600 : \n";
    MyQueue.Print();

    //Extension #4
    MyQueue.InsertAfter(2, 800);
    cout << "\n\nQueue after Inserting 800 after Item(2) : \n";
    MyQueue.Print();


    //Extension #5
    MyQueue.InsertAtFront(1000);
    cout << "\n\nQueue after Inserting 1000 at front: \n";
    MyQueue.Print();

    //Extension #6
    MyQueue.InsertAtBack(2000);
    cout << "\n\nQueue after Inserting 2000 at back: \n";
    MyQueue.Print();

    //Extension #7
```

```
    MyQueue.Clear();
    cout << "\n\nQueue after Clear(): \n";
    MyQueue.Print();



    system("pause>0");

}
```

Project 3

Requirements

عاوزين نعمل كلاس لل stack بنفس ال functions اللي عملناها في ال queue بس هنعملها في ال stack

```
#include <iostream>
#include "clsMyStack.h"

using namespace std;

int main()
{

    clsMyStack <int> MyStack;

    MyStack.push(10);
    MyStack.push(20);
    MyStack.push(30);
    MyStack.push(40);
    MyStack.push(50);


    cout << "\nStack: \n";
    MyStack.Print();

    cout << "\nStack Size: " << MyStack.Size();
```

```
cout << "\nStack Top: " << MyStack.Top();
cout << "\nStack Bottom: " << MyStack.Bottom();
(const char [16])"\nStack Bottom: "
Search Online

MyStack.pop();

cout << "\n\nStack after pop() : \n";
MyStack.Print();

//Extension #1
cout << "\n\n Item(2) : " << MyStack.GetItem(2);

//Extension #2
MyStack.Reverse();
cout << "\n\nStack after reverse() : \n";
MyStack.Print();
```

```
//Extension #3
MyStack.UpdateItem(2, 600);
cout << "\n\nStack after updating Item
MyStack.Print();

//Extension #4
MyStack.InsertAfter(2, 800);
cout << "\n\nStack after Inserting 800
MyStack.Print();


//Extension #5
MyStack.InsertAtFront(1000);
cout << "\n\nStack after Inserting 100
MyStack.Print();

//Extension #6
MyStack.InsertAtBack(2000);
cout << "\n\nStack after Inserting 200
MyStack.Print();

//Extension #7
MyStack.Clear();
cout << "\n\nStack after Clear() : \n";
```



```
Stack:
50 40 30 20 10

Stack Size: 5
Stack Top: 50
Stack Bottom: 10

Stack after pop() :
40 30 20 10


 Item(2) : 20

Stack after reverse() :
10 20 30 40


Stack after updating Item(2) to 600 :
10 20 600 40


Stack after Inserting 800 after Item(2) :
10 20 600 800 40


Stack after Inserting 1000 at top:
1000 10 20 600 800 40
```

```
Stack after Inserting 2000 at bottom:
1000 10 [illegible] 600 800 40 2000

Stack after Clear():
```

Solution

الفرق بين ال queue وال stack هوا ان ال queue بيعمل push في النهايه وال stack بيعمله في البدايه

عشان كده هوا خلي ال stack يورث من ال queue ويعمل بس override لل functions المختلفه

زي ال push خلاها تضيف في البدايه

وال top وال bottom بس عشان يغير الاسم

وده كود الكلاس كله

```
#pragma once

#include <iostream>
#include "clsMyQueue.h"
using namespace std;
template <class T>

class  clsMyStack :public   clsMyQueue <T>
{

public:

       void push(T Item)
       {
              clsMyQueue <T>::_MyList.InsertAtBeginning(Item);
       }

       T Top()
       {
              return clsMyQueue <T>::front();

       }

       T Bottom()
       {
              return clsMyQueue <T>::back();

       }

};
```

وده ال main

```
//ProgrammingAdvices.com
//Mohammed Abu-Hadhoud

#include <iostream>
#include "clsMyStack.h"

using namespace std;

int main()
{

    clsMyStack <int> MyStack;
```

```
    MyStack.push(10);
    MyStack.push(20);
    MyStack.push(30);
    MyStack.push(40);
    MyStack.push(50);


    cout << "\nStack: \n";
    MyStack.Print();

    cout << "\nStack Size: " << MyStack.Size();
    cout << "\nStack Top: " << MyStack.Top();
    cout << "\nStack Bottom: " << MyStack.Bottom();

    MyStack.pop();

    cout << "\n\nStack after pop() : \n";
    MyStack.Print();

    //Extension #1
    cout << "\n\n Item(2) : " << MyStack.GetItem(2);

    //Extension #2
    MyStack.Reverse();
    cout << "\n\nStack after reverse() : \n";
    MyStack.Print();

    //Extension #3
    MyStack.UpdateItem(2, 600);
    cout << "\n\nStack after updating Item(2) to 600 : \n";
    MyStack.Print();

    //Extension #4
    MyStack.InsertAfter(2, 800);
    cout << "\n\nStack after Inserting 800 after Item(2) : \n";
    MyStack.Print();


    //Extension #5
    MyStack.InsertAtFront(1000);
    cout << "\n\nStack after Inserting 1000 at top: \n";
    MyStack.Print();

    //Extension #6
    MyStack.InsertAtBack(2000);
    cout << "\n\nStack after Inserting 2000 at bottom: \n";
    MyStack.Print();

    //Extension #7
    MyStack.Clear();
    cout << "\n\nStack after Clear(): \n";
    MyStack.Print();

    system("pause>0");

}
```

Project 4

Requirements

عاوزين نعمل my dynamic array

```
#include <iostream>
#include "clsDynamicArray.h"

using namespace std;

int main()
{

    clsDynamicArray <int> MyDynamicArray(5);

    MyDynamicArray.SetItem(0, 10);
    MyDynamicArray.SetItem(1, 20);
    MyDynamicArray.SetItem(2, 30);
    MyDynamicArray.SetItem(3, 40);
    MyDynamicArray.SetItem(4, 50);

    cout << "\nIs Empty?  " << MyDynamicArray.IsEmpty();
    cout << "\nArray Size: " << MyDynamicArray.Size() << "\n";
    cout << "\nArray Items: \n";

    MyDynamicArray.PrintList();
```

```
Is Empty?  0
Array Size: 5

Array Items:
10 20 30 40 50
```

```
MyDynamicArray(10);
```

```
Is Empty?  0
Array Size: 10

Array Items:
10 20 30 40 50 -842150451 -842150451 -842150451 -842150451 -842150451
```

Solution

عشان تفهم اكتر راجع ال pointer في الكورس السادس

ده كود الكلاس

```
#pragma once

#include <iostream>
using namespace std;

template <class T>
class clsDynamicArray
{

protected:
    int _Size = 0;


public:
    T* OriginalArray;
```

```
    clsDynamicArray(int Size = 0)
    {
        if (Size < 0)
            Size = 0;

        _Size = Size;

        OriginalArray = new T[_Size];

    }

    ~clsDynamicArray()
    {

        delete[]  OriginalArray;

    }

    bool SetItem(int index, T Value)
    {

        if (index >= _Size || _Size < 0)
        {
            return false;
        }

        OriginalArray[index] = Value;
        return true;

    }


    int Size()
    {
        return _Size;
    }

    bool IsEmpty()
    {
        return (_Size == 0 ? true : false);

    }

    void PrintList()

    {

        for (int i = 0; i <= _Size - 1; i++)
        {
            cout << OriginalArray[i] << " ";
        }

        cout << "\n";

    }



};
```

وده كود ال main

```
//ProgrammingAdvices.com
//Mohammed Abu-Hadhoud

#include <iostream>
#include "clsDynamicArray.h"

using namespace std;
```

```
int main()
{

    clsDynamicArray <int> MyDynamicArray(5);

    MyDynamicArray.SetItem(0, 10);
    MyDynamicArray.SetItem(1, 20);
    MyDynamicArray.SetItem(2, 30);
    MyDynamicArray.SetItem(3, 40);
    MyDynamicArray.SetItem(4, 50);

    cout << "\nIs Empty?  " << MyDynamicArray.IsEmpty();
    cout << "\nArray Size: " << MyDynamicArray.Size() << "\n";
    cout << "\nArray Items: \n";

    MyDynamicArray.PrintList();





    system("pause>0");

}
```

Extension 01 - Requirements

عاوزين نغير حجم ال array ياما نكبرها او نصغرهابحيث اني لو صغرتها الداتا اللي فيها ماتروحش (قصده العناصر اللي فضلت موجوده انما الباقي هيتشال اكيد) ولوكبرتها هتفضل الداتا زي ما هيا وباقي العناصر فاضيين

```
MyDynamicArray.PrintList();


MyDynamicArray.Resize(2);
cout << "\nArray Size: " << MyDynamicArray.Size() << "\n";
cout << "\nArray Items after resize to 2 : \n";
MyDynamicArray.PrintList();
```



```
MyDynamicArray.Resize(10);
cout << "\nArray Size: " << MyDynamicArray.Size() << "\n";
cout << "\nArray Items after resize to 10 : \n";
MyDynamicArray.PrintList();
```

```
Is Empty?  0
Array Size: 5

Array Items:
10 20 30 40 50

Array Size: 2

Array Items after resize to 2 :
10 20

Array Size: 10

Array Items after resize to 10 :
10 20 -842150451 -842150451 -842150451 -842150451 -842150451 -842150451 -842150451 -842150451
```

Extension 01 - Solution

ده كود الكلاس

```
#pragma once

#include <iostream>
using namespace std;

template <class T>
class clsDynamicArray
{

protected:
    int _Size = 0;
    T* _TempArray;

public:
    T* OriginalArray;

    clsDynamicArray(int Size = 0)
    {
        if (Size < 0)
            Size = 0;

        _Size = Size;

        OriginalArray = new T[_Size];

    }

    ~clsDynamicArray()
    {

        delete[]  OriginalArray;

    }

    bool SetItem(int index, T Value)
    {

        if (index >= _Size || _Size < 0)
        {
            return false;
        }

        OriginalArray[index] = Value;
        return true;

    }


    int Size()
    {
        return _Size;
    }

    bool IsEmpty()
    {
        return (_Size == 0 ? true : false);

    }

    void PrintList()

    {

        for (int i = 0; i <= _Size - 1; i++)
        {
            cout << OriginalArray[i] << " ";
        }
```

```
        cout << "\n";

    }

    void Resize(int NewSize)
    {

        if (NewSize < 0)
        {
            NewSize = 0;
        };

        _TempArray = new T[NewSize];

        //limit the original size to the new size if it is less.
        if (NewSize < _Size)
            _Size = NewSize;

        //copy all data from original array until the size
        for (int i = 0; i < _Size; i++)
        {
            _TempArray[i] = OriginalArray[i];
        }

        _Size = NewSize;

        delete[] OriginalArray;
        OriginalArray = _TempArray;

    }


};
```

وده ال main

```
//ProgrammingAdvices.com
//Mohammed Abu-Hadhoud

#include <iostream>
#include "clsDynamicArray.h"

using namespace std;

int main()
{

    clsDynamicArray <int> MyDynamicArray(5);

    MyDynamicArray.SetItem(0, 10);
    MyDynamicArray.SetItem(1, 20);
    MyDynamicArray.SetItem(2, 30);
    MyDynamicArray.SetItem(3, 40);
    MyDynamicArray.SetItem(4, 50);

    cout << "\nIs Empty?  " << MyDynamicArray.IsEmpty();
    cout << "\nArray Size: " << MyDynamicArray.Size() << "\n";
    cout << "\nArray Items: \n";

    MyDynamicArray.PrintList();


    MyDynamicArray.Resize(2);
    cout << "\nArray Size: " << MyDynamicArray.Size() << "\n";
    cout << "\nArray Items after resize to 2 : \n";
    MyDynamicArray.PrintList();



    MyDynamicArray.Resize(10);
    cout << "\nArray Size: " << MyDynamicArray.Size() << "\n";
```

```
    cout << "\nArray Items after resize to 10 : \n";
    MyDynamicArray.PrintList();


    system("pause>0");



}
```

Extension 02 to 04 - Requirements

عاوزين نعمل reverse و get item و clear

```
cout << "\nItem(2): " << MyDynamicArray.GetItem(2) << "\n";


MyDynamicArray.Reverse();

cout << "\nArray Items after reverse: \n";
MyDynamicArray.PrintList();

MyDynamicArray.Clear();

cout << "\nArray Items after clear: \n";
MyDynamicArray.PrintList();
```

```
Is Empty?  0
Array Size: 5

Array Items:
10 20 30 40 50

Item(2): 30

Array Items after reverse:
50 40 30 20 10

Array Items after clear:
```

Extension 02 to 04 - Solution

ده التعديل عالكلاس

```
    T GetItem(int index)
    {
        return OriginalArray[index];

    }


    void  Reverse()
    {

        _TempArray = new T[_Size];

        int counter = 0;

        for (int i = _Size - 1; i >= 0; i--)
        {
            _TempArray[counter] = OriginalArray[i];
            counter++;
        }
```

```
        delete[] OriginalArray;
        OriginalArray = _TempArray;

    }


    void Clear()
    {
        _Size = 0;
        _TempArray = new T[0];
        delete[] OriginalArray;
        OriginalArray = _TempArray;
    }
```

وده ال main

```
//ProgrammingAdvices.com
//Mohammed Abu-Hadhoud

#include <iostream>
#include "clsDynamicArray.h"

using namespace std;

int main()
{

    clsDynamicArray <int> MyDynamicArray(5);

    MyDynamicArray.SetItem(0, 10);
    MyDynamicArray.SetItem(1, 20);
    MyDynamicArray.SetItem(2, 30);
    MyDynamicArray.SetItem(3, 40);
    MyDynamicArray.SetItem(4, 50);

    cout << "\nIs Empty?  " << MyDynamicArray.IsEmpty();
    cout << "\nArray Size: " << MyDynamicArray.Size() << "\n";
    cout << "\nArray Items: \n";

    MyDynamicArray.PrintList();

    cout << "\nItem(2): " << MyDynamicArray.GetItem(2) << "\n";


    MyDynamicArray.Reverse();

    cout << "\nArray Items after reverse: \n";
    MyDynamicArray.PrintList();

    MyDynamicArray.Clear();

    cout << "\nArray Items after clear: \n";
    MyDynamicArray.PrintList();

    system("pause>0");


}
```

Extension 05 - Requirements

عاوزين نعمل delete item عن طريق انك تديله ال index

```
MyDynamicArray.DeleteItemAt(2);
cout << "\nArray Items after deleting item(2): \n";
cout << "\nArray Size: " << MyDynamicArray.Size() << "\n";
MyDynamicArray.PrintList();
```

Extension 05 - Solution

ده التعديل علي الكلاس

```
    bool DeleteItemAt(int index)
    {

        if (index >= _Size || index < 0)
        {
            return false;
        }

        _Size--;

        _TempArray = new T[_Size];

        //copy all before index
        for (int i = 0; i < index; i++)
        {
            _TempArray[i] = OriginalArray[i];
        }

        //copy all after index
        for (int i = index + 1; i < _Size + 1; i++)
        {
            _TempArray[i - 1] = OriginalArray[i];
        }

        delete[] OriginalArray;
        OriginalArray = _TempArray;
        return true;

    }
```

ده ال main

```
//ProgrammingAdvices.com
//Mohammed Abu-Hadhoud

#include <iostream>
#include "clsDynamicArray.h"

using namespace std;

int main()
{

    clsDynamicArray <int> MyDynamicArray(5);

    MyDynamicArray.SetItem(0, 10);
    MyDynamicArray.SetItem(1, 20);
    MyDynamicArray.SetItem(2, 30);
    MyDynamicArray.SetItem(3, 40);
    MyDynamicArray.SetItem(4, 50);

    cout << "\nIs Empty?  " << MyDynamicArray.IsEmpty();
    cout << "\nArray Size: " << MyDynamicArray.Size() << "\n";
    cout << "\nArray Items: \n";

    MyDynamicArray.PrintList();

    MyDynamicArray.DeleteItemAt(2);
    cout << "\nArray Items after deleting item(2): \n";
    cout << "\nArray Size: " << MyDynamicArray.Size() << "\n";
    MyDynamicArray.PrintList();

    system("pause>0");

}
```

Extension 06 to 07 - Requirements

عاوزين نعمل delete first item و delete last item

```
MyDynamicArray.DeleteFirstItem();
cout << "\nArray Items after deleting FirstItem: \n";
cout << "\nArray Size: " << MyDynamicArray.Size() << "\n";
MyDynamicArray.PrintList();

MyDynamicArray.DeleteLastItem();
cout << "\nArray Items after deleting LastItem: \n";
cout << "\nArray Size: " << MyDynamicArray.Size() << "\n";
MyDynamicArray.PrintList();
```

Extension 06 to 07 - Solution

ده التعديل

```
    void DeleteFirstItem()
    {

        DeleteItemAt(0);

    }

    void DeleteLastItem()
    {

        DeleteItemAt(_Size - 1);
    }
```

وده ال main

```
//ProgrammingAdvices.com
//Mohammed Abu-Hadhoud

#include <iostream>
#include "clsDynamicArray.h"

using namespace std;

int main()
{

    clsDynamicArray <int> MyDynamicArray(5);

    MyDynamicArray.SetItem(0, 10);
    MyDynamicArray.SetItem(1, 20);
    MyDynamicArray.SetItem(2, 30);
    MyDynamicArray.SetItem(3, 40);
    MyDynamicArray.SetItem(4, 50);

    cout << "\nIs Empty?  " << MyDynamicArray.IsEmpty();
    cout << "\nArray Size: " << MyDynamicArray.Size() << "\n";
    cout << "\nArray Items: \n";

    MyDynamicArray.PrintList();

    MyDynamicArray.DeleteFirstItem();
    cout << "\nArray Items after deleting FirstItem: \n";
    cout << "\nArray Size: " << MyDynamicArray.Size() << "\n";
    MyDynamicArray.PrintList();
```

```
    MyDynamicArray.DeleteLastItem();
    cout << "\nArray Items after deleting LastItem: \n";
    cout << "\nArray Size: " << MyDynamicArray.Size() << "\n";
    MyDynamicArray.PrintList();

    system("pause>0");

}
```

Extension 08 to 09 - Requirements

عاوزين نعمل find بس يدور عال value]d,ويجيبلك index وعاوزين نعمل delete بال value برضه

```
int Index = MyDynamicArray.Find(30);
if (Index == -1)
    cout << "\nItem was not Found :-(\n ";
else
    cout << "\n30 is found at index : " << Index;

MyDynamicArray.DeleteItem(30);
cout << "\n\nArray Items after deleting 30:";
cout << "\nArray Size: " << MyDynamicArray.Size() << "\n";
MyDynamicArray.PrintList();
```



```
Is Empty?  0
Array Size: 5

Array Items:
10 20 30 40 50

30 is found at index : 2

Array Items after deleting 30:
Array Size: 4
10 20 40 50
```

Extension 08 to 09 - Solution

ده التعديل

```
int Find(T Value)
{
    for (int i = 0; i < _Size; i++)
    {
        if (OriginalArray[i] == Value)
        {
            return i;
        }
    }
    return -1;

}
```

```
    bool DeleteItem(T Value) {

        int index = Find(Value);

        if (index == -1)
        {
            return false;
        }

        DeleteItemAt(index);
        return true;

    }
```

وده ال main

```
//ProgrammingAdvices.com
//Mohammed Abu-Hadhoud

#include <iostream>
#include "clsDynamicArray.h"

using namespace std;

int main()
{

    clsDynamicArray <int> MyDynamicArray(5);

    MyDynamicArray.SetItem(0, 10);
    MyDynamicArray.SetItem(1, 20);
    MyDynamicArray.SetItem(2, 30);
    MyDynamicArray.SetItem(3, 40);
    MyDynamicArray.SetItem(4, 50);

    cout << "\nIs Empty?  " << MyDynamicArray.IsEmpty();
    cout << "\nArray Size: " << MyDynamicArray.Size() << "\n";
    cout << "\nArray Items: \n";

    MyDynamicArray.PrintList();

    int Index = MyDynamicArray.Find(30);
    if (Index == -1)
        cout << "\nItem was not Found :-(\n ";
    else
        cout << "\n30 is found at index : " << Index;

    MyDynamicArray.DeleteItem(30);
    cout << "\n\nArray Items after deleting 30:";
    cout << "\nArray Size: " << MyDynamicArray.Size() << "\n";
    MyDynamicArray.PrintList();


    system("pause>0");

}
```

Extension 10 - Requirements

عاوزين نعمل insert at تديها ال index تحط فيه العنصر

```
MyDynamicArray.InsertAt(2, 500);
cout << "\n\nArray after insert 500 at index 2:";
cout << "\nArray Size: " << MyDynamicArray.Size() << "\n";
MyDynamicArray.PrintList();
```

```
Is Empty?  0
Array Size: 5

Array Items:
10 20 30 40 50

Array after insert 500 at index 2:
Array Size: 6
10 20 500 30 40 50
```

Extension 10 - Solution

ده التعديل علي الكلاس

```
    bool InsertAt(T index, T value) {

        if (index > _Size || index < 0)
        {
            return false;
        }

        _Size++;

        _TempArray = new T[_Size];

        //copy all before index
        for (int i = 0; i < index; i++)
        {
            _TempArray[i] = OriginalArray[i];
        }

        _TempArray[index] = value;

        //copy all after index
        for (int i = index; i < _Size - 1; i++)
        {
            _TempArray[i + 1] = OriginalArray[i];
        }

        delete[] OriginalArray;
        OriginalArray = _TempArray;
        return true;

    }
```

وده ال main

```
//ProgrammingAdvices.com
//Mohammed Abu-Hadhoud

#include <iostream>
#include "clsDynamicArray.h"

using namespace std;

int main()
{

    clsDynamicArray <int> MyDynamicArray(5);

    MyDynamicArray.SetItem(0, 10);
```

```
    MyDynamicArray.SetItem(1, 20);
    MyDynamicArray.SetItem(2, 30);
    MyDynamicArray.SetItem(3, 40);
    MyDynamicArray.SetItem(4, 50);

    cout << "\nIs Empty?  " << MyDynamicArray.IsEmpty();
    cout << "\nArray Size: " << MyDynamicArray.Size() << "\n";
    cout << "\nArray Items: \n";

    MyDynamicArray.PrintList();

    MyDynamicArray.InsertAt(2, 500);
    cout << "\n\nArray after insert 500 at index 2:";
    cout << "\nArray Size: " << MyDynamicArray.Size() << "\n";
    MyDynamicArray.PrintList();


    system("pause>0");

}
```

Extension 11 to 14 - Requirements

عاوزين نعمل insert at beginning و insert after و insert at end و insert before

```
MyDynamicArray.InsertAtBeginning(400);
cout << "\n\nArray after insert 400 at Begining:";
cout << "\nArray Size: " << MyDynamicArray.Size() << "\n";
MyDynamicArray.PrintList();

MyDynamicArray.InsertBefore(2, 500);
cout << "\n\nArray after insert 500 before index 2:";
cout << "\nArray Size: " << MyDynamicArray.Size() << "\n";
MyDynamicArray.PrintList();
```

```
MyDynamicArray.InsertAfter(2, 600);
cout << "\n\nArray after insert 600 after index 2:";
cout << "\nArray Size: " << MyDynamicArray.Size() << "\n";
MyDynamicArray.PrintList();


MyDynamicArray.InsertAtEnd(800);
cout << "\n\nArray after insert 800 at End:";
cout << "\nArray Size: " << MyDynamicArray.Size() << "\n";
MyDynamicArray.PrintList();
```

Extension 11 to 14 - Solution

ده التعديل علي الكلاس

```
    void InsertAtBeginning(T value)
    {

        InsertAt(0, value);

    }
```

```
    bool InsertBefore(T index, T value)
    {
        if (index < 1)
            return InsertAt(0, value);
        else
            return InsertAt(index - 1, value);

    }

    bool InsertAfter(T index, T value)
    {
        if (index >= _Size)
            return InsertAt(_Size - 1, value);
        else
            return InsertAt(index + 1, value);

    }

    bool InsertAtEnd(T value)
    {

        return InsertAt(_Size, value);

    }
```

وده كود ال main

```
//ProgrammingAdvices.com
//Mohammed Abu-Hadhoud

#include <iostream>
#include "clsDynamicArray.h"

using namespace std;

int main()
{

    clsDynamicArray <int> MyDynamicArray(5);

    MyDynamicArray.SetItem(0, 10);
    MyDynamicArray.SetItem(1, 20);
    MyDynamicArray.SetItem(2, 30);
    MyDynamicArray.SetItem(3, 40);
    MyDynamicArray.SetItem(4, 50);

    cout << "\nIs Empty?  " << MyDynamicArray.IsEmpty();
    cout << "\nArray Size: " << MyDynamicArray.Size() << "\n";
    cout << "\nArray Items: \n";

    MyDynamicArray.PrintList();

    MyDynamicArray.InsertAtBeginning(400);
    cout << "\n\nArray after insert 400 at Begining:";
    cout << "\nArray Size: " << MyDynamicArray.Size() << "\n";
    MyDynamicArray.PrintList();

    MyDynamicArray.InsertBefore(2, 500);
    cout << "\n\nArray after insert 500 before index 2:";
    cout << "\nArray Size: " << MyDynamicArray.Size() << "\n";
    MyDynamicArray.PrintList();

    MyDynamicArray.InsertAfter(2, 600);
    cout << "\n\nArray after insert 600 after index 2:";
    cout << "\nArray Size: " << MyDynamicArray.Size() << "\n";
    MyDynamicArray.PrintList();


    MyDynamicArray.InsertAtEnd(800);
    cout << "\n\nArray after insert 800 at End:";
```

```
    cout << "\nArray Size: " << MyDynamicArray.Size() << "\n";
    MyDynamicArray.PrintList();


    system("pause>0");

}
```

Project 5

Requirements

عاوزين نعمل my queue بيعتمد علي ال array اللي احنا عاملينه

بنفس ال functions اللي كنا عاملينها

```
clsMyQueueArr <int> MyQueue;

MyQueue.push(10);
MyQueue.push(20);
MyQueue.push(30);
MyQueue.push(40);
MyQueue.push(50);


cout << "\nQueue: \n";
MyQueue.Print();

cout << "\nQueue Size: " << MyQueue.Size();
cout << "\nQueue Front: " << MyQueue.front();
cout << "\nQueue Back: " << MyQueue.back();

MyQueue.pop();

cout << "\n\nQueue after pop() : \n";
MyQueue.Print();
```

```
MyQueue.Reverse();
cout << "\n\nQueue after reverse() : \n";
MyQueue.Print();


MyQueue.UpdateItem(2, 600);
cout << "\n\nQueue after updating Item(2) to 600 : \n";
MyQueue.Print();


MyQueue.InsertAfter(2, 800);
cout << "\n\nQueue after Inserting 800 after Item(2) : \n";
MyQueue.Print();



MyQueue.InsertAtFront(1000);
cout << "\n\nQueue after Inserting 1000 at front: \n";
MyQueue.Print();


MyQueue.InsertAtBack(2000);
cout << "\n\nQueue after Inserting 2000 at back: \n";
MyQueue.Print();


MyQueue.Clear();
```



```
Queue:
10 20 30 40 50

Queue Size: 5
Queue Front: 10
Queue Back: 50

Queue after pop() :
20 30 40 50


 Item(2) : 40

Queue after reverse() :
50 40 30 20


Queue after updating Item(2) to 600 :
50 40 600 20


Queue after Inserting 800 after Item(2) :
50 40 600 800 20


Queue after Inserting 1000 at front:
1000 50 40 600 800 20
```

```
Queue after Inserting 2000 at back:
1000 50 40 600 800 20 2000


Queue after Clear():
```

Solution

ده كود الكلاس

```
#pragma once

#include <iostream>
#include "clsDynamicArray.h"


using namespace std;
template <class T>

class clsMyQueueArr
{

protected:
	clsDynamicArray <T> _MyList;

public:

	void push(T Item)
	{
		_MyList.InsertAtEnd(Item);
	}


	void pop()
	{//*
		_MyList.DeleteFirstItem();
	}

	void Print()
	{
		_MyList.PrintList();
	}

	int Size()
	{
		return _MyList.Size();
	}

	bool IsEmpty()
	{
		return _MyList.IsEmpty();
	}

	T front()
	{
		return _MyList.GetItem(0);
	}

	T back()
	{
		return _MyList.GetItem(Size() - 1);
	}

	T GetItem(int Index)
	{
		return  _MyList.GetItem(Index);
```

```
        }

        void Reverse()
        {

                _MyList.Reverse();

        }

        void UpdateItem(int Index, T NewValue)
        {
                _MyList.SetItem(Index, NewValue);

        }

        void InsertAfter(int Index, T NewValue)
        {
                _MyList.InsertAfter(Index, NewValue);
        }


        void InsertAtFront(T Value)
        {
                _MyList.InsertAtBeginning(Value);

        }

        void InsertAtBack(T Value)
        {
                _MyList.InsertAtEnd(Value);


        }


        void Clear()
        {
                _MyList.Clear();
        }



};
```

وده ال main

```
//ProgrammingAdvices.com
//Mohammed Abu-Hadhoud

#include <iostream>
#include "clsMyQueueArr.h"

using namespace std;

int main()
{

    clsMyQueueArr <int> MyQueue;

    MyQueue.push(10);
    MyQueue.push(20);
    MyQueue.push(30);
    MyQueue.push(40);
    MyQueue.push(50);


    cout << "\nQueue: \n";
    MyQueue.Print();

    cout << "\nQueue Size: " << MyQueue.Size();
```

```
    cout << "\nQueue Front: " << MyQueue.front();
    cout << "\nQueue Back: " << MyQueue.back();

    MyQueue.pop();

    cout << "\n\nQueue after pop() : \n";
    MyQueue.Print();


    cout << "\n\n Item(2) : " << MyQueue.GetItem(2);


    MyQueue.Reverse();
    cout << "\n\nQueue after reverse() : \n";
    MyQueue.Print();


    MyQueue.UpdateItem(2, 600);
    cout << "\n\nQueue after updating Item(2) to 600 : \n";
    MyQueue.Print();


    MyQueue.InsertAfter(2, 800);
    cout << "\n\nQueue after Inserting 800 after Item(2) : \n";
    MyQueue.Print();



    MyQueue.InsertAtFront(1000);
    cout << "\n\nQueue after Inserting 1000 at front: \n";
    MyQueue.Print();


    MyQueue.InsertAtBack(2000);
    cout << "\n\nQueue after Inserting 2000 at back: \n";
    MyQueue.Print();


    MyQueue.Clear();
    cout << "\n\nQueue after Clear(): \n";
    MyQueue.Print();

    system("pause>0");


}
```

Requirements

عاوزين نعمل my stack arr بيعتمد علي ال array مش علي ال linked list بنفس الطريقه اللي عملناها المره اللي فاتت بال inheritance

Solution

ده كود الكلاس

```
#pragma once

#include <iostream>
#include "clsMyQueueArr.h"
using namespace std;
template <class T>
```

```
class  clsMyStackArr :public   clsMyQueueArr <T>
{

public:

      void push(T Item)
      {
            clsMyQueueArr <T>::_MyList.InsertAtBeginning(Item);
      }

      T Top()
      {
            return clsMyQueueArr <T>::front();

      }

      T Bottom()
      {
            return clsMyQueueArr <T>::back();

      }

};
```

وده ال main

```
//ProgrammingAdvices.com
//Mohammed Abu-Hadhoud

#include <iostream>
#include "clsMyStackArr.h"

using namespace std;

int main()
{

    clsMyStackArr <int> MyStack;

    MyStack.push(10);
    MyStack.push(20);
    MyStack.push(30);
    MyStack.push(40);
    MyStack.push(50);


    cout << "\nStack: \n";
    MyStack.Print();

    cout << "\nStack Size: " << MyStack.Size();
    cout << "\nStack Top: " << MyStack.Top();
    cout << "\nStack Bottom: " << MyStack.Bottom();

    MyStack.pop();

    cout << "\n\nStack after pop() : \n";
    MyStack.Print();


    cout << "\n\n Item(2) : " << MyStack.GetItem(2);


    MyStack.Reverse();
    cout << "\n\nStack after reverse() : \n";
    MyStack.Print();


    MyStack.UpdateItem(2, 600);
    cout << "\n\nStack after updating Item(2) to 600 : \n";
    MyStack.Print();
```

```
    MyStack.InsertAfter(2, 800);
    cout << "\n\nStack after Inserting 800 after Item(2) : \n";
    MyStack.Print();



    MyStack.InsertAtFront(1000);
    cout << "\n\nStack after Inserting 1000 at top: \n";
    MyStack.Print();


    MyStack.InsertAtBack(2000);
    cout << "\n\nStack after Inserting 2000 at bottom: \n";
    MyStack.Print();


    MyStack.Clear();
    cout << "\n\nStack after Clear(): \n";
    MyStack.Print();

    system("pause>0");

}
```

Project 7

Requirements

المشروع تطبيق عملي علي ال call stack ومن التطبيقات عال stack هيا انه function تستدعي واحده تانيه وحوار ال undo اللي في ال word او الاكسيل

فاحنا هنعمل simulation لموضوع ال undo وال redo

```
#include <iostream>
#include "clsMyString.h"

using namespace std;

int main()
{

    cout << "\n\n\t\t\t\t\t\t Undo/Redo Project\n\n";

    clsMyString S1;

    cout << "\nS1  = " << S1.Value << "\n";

    S1.Value="Mohammed";

    cout << "S1  = " << S1.Value << "\n";

    S1.Value = "Mohammed2";
```

```
cout << "S1  = " << S1.Value << "\n";

S1.Value = "Mohammed3";

cout << "S1  = " << S1.Value << "\n";

cout << "\n\nUndo: ";
cout << "\n_________\n";

S1.Undo();
cout << "\nS1  after undo = " << S1.Value << "\n";

S1.Undo();
cout << "S1  after undo = " << S1.Value << "\n";

S1.Undo();
cout << "S1  after undo = " << S1.Value << "\n";
```



```
cout << "\n\nRedo: ";
cout << "\n_________\n";

S1.Redo();
cout << "\nS1  after Redo = " << S1.Value << "\n";

S1.Redo();
cout << "S1  after Redo = " << S1.Value << "\n";

S1.Redo();
cout << "S1  after Redo = " << S1.Value << "\n";
```

```
Undo/Redo Project

S1  =
S1  = Mohammed
S1  = Mohammed2
S1  = Mohammed3

Undo:
_________

S1  after undo = Mohammed2
S1  after undo = Mohammed
S1  after undo =

Redo:
_________

S1  after Redo = Mohammed
S1  after Redo = Mohammed2
S1  after Redo = Mohammed3
```

Solution

ده كود الكلاس

```
#pragma once

#include <stack>

using namespace std;

class clsMyString
{

private:

	stack <string> _Undo;
	stack <string> _Redo;
	string _Value;

public:

	void Set(string value)
	{
		_Undo.push(_Value);
		_Value = value;
	}

	string Get()
	{
		return _Value;
	}

	__declspec(property(get = Get, put = Set)) string Value;


	void Undo()
	{
```

```
            if (!_Undo.empty())
            {
                _Redo.push(_Value);
                _Value = _Undo.top();
                _Undo.pop();
            }

        }

        void Redo()
        {

            if (!_Redo.empty())
            {
                _Undo.push(_Value);
                _Value = _Redo.top();
                _Redo.pop();
            }


        }
};
```

وده الmain

```
//ProgrammingAdvices.com
//Mohammed Abu-Hadhoud

#include <iostream>
#include "clsMyString.h"

using namespace std;

int main()
{

    cout << "\n\n\t\t\t\t\t\t Undo/Redo Project\n\n";

    clsMyString S1;

    cout << "\nS1  = " << S1.Value << "\n";

    S1.Value = "Mohammed";

    cout << "S1  = " << S1.Value << "\n";

    S1.Value = "Mohammed2";

    cout << "S1  = " << S1.Value << "\n";

    S1.Value = "Mohammed3";

    cout << "S1  = " << S1.Value << "\n";

    cout << "\n\nUndo: ";
    cout << "\n__________\n";

    S1.Undo();
    cout << "\nS1  after undo = " << S1.Value << "\n";

    S1.Undo();
    cout << "S1  after undo = " << S1.Value << "\n";

    S1.Undo();
    cout << "S1  after undo = " << S1.Value << "\n";

    cout << "\n\nRedo: ";
    cout << "\n__________\n";

    S1.Redo();
    cout << "\nS1  after Redo = " << S1.Value << "\n";
```

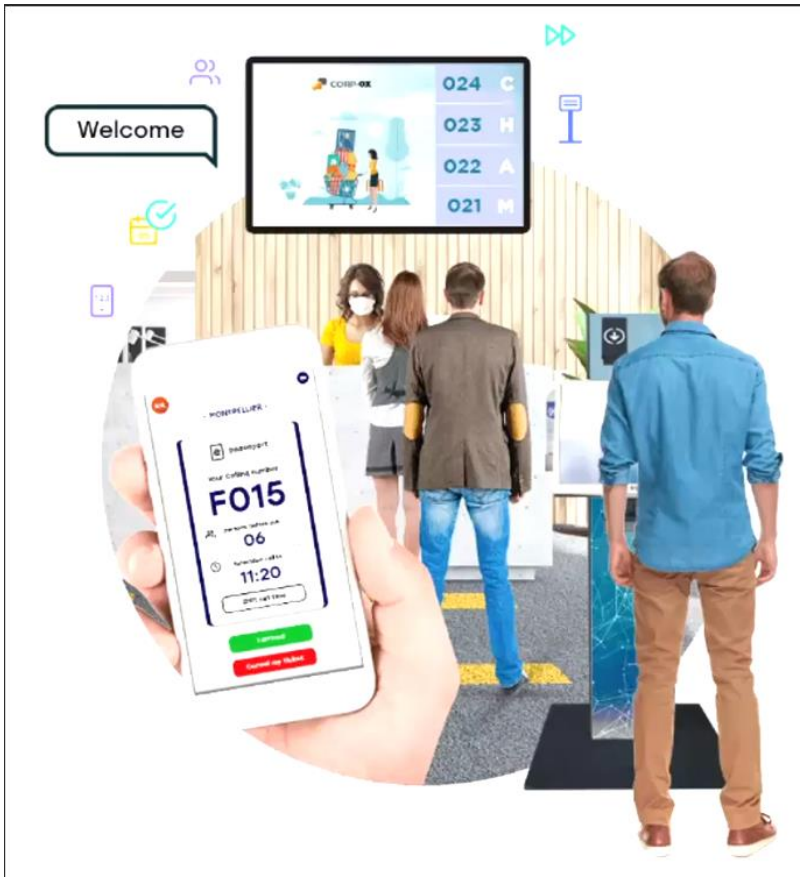
```
    S1.Redo();
    cout << "S1  after Redo = " << S1.Value << "\n";

    S1.Redo();
    cout << "S1  after Redo = " << S1.Value << "\n";

    system("pause>0");

    return 0;
}
```

Project 8

Requirements

هنا المشروع ده تطبيق علي ال queue هنعمل queue line مبني علي ال queue data structure ومن استخداماته شاشات الانتظار بتاعت البنوك او فروع شركات الاتصالات لما تيجي تدخل بتقف علي شاشه وبتختار مثلا انك تشتري خط جديد او تشحن رصيدك او أي حاجه تانيه

كل خيار من دول هوا عباره عن queue line



```
#include <iostream>
#include "clsQueueLine.h"

using namespace std;

int main()
{

    clsQueueLine PayBillsQueue("A0", 10);
    clsQueueLine SubscriptionsQueue("B0", 5);

    PayBillsQueue.IssueTicket();
    PayBillsQueue.IssueTicket();
    PayBillsQueue.IssueTicket();
    PayBillsQueue.IssueTicket();
    PayBillsQueue.IssueTicket();

    cout << "\nPay Bills Queue Info:\n";
    PayBillsQueue.PrintInfo();
```

```
PayBillsQueue.PrintTicketsLineRTL();
PayBillsQueue.PrintTicketsLineLTR();
```

```
PayBillsQueue.PrintAllTickets();
```

```
Pay Bills Queue Info:

_____________________________

          Queue Info

_____________________________

   Prefix    = A0
   Total Tickets    = 5
   Served Clients   = 0
   Wating Clients   = 5

_____________________________

Tickets:  A01 <--  A02 <--  A03 <--  A04 <--  A05 <--

Tickets:  A05 -->  A04 -->  A03 -->  A02 -->  A01 -->

          ---Tickets---

_____________________________

              A01

   4/2/2023 - 15:11:8
   Wating Clients = 0
```

```
---Tickets---

A01

4/2/2023 - 15:11:8
Wating Clients = 0
Serve Time In
0 Minutes.

A02

4/2/2023 - 15:11:8
Wating Clients = 1
Serve Time In
10 Minutes.

A03
```

التاريخ اللي في التذكره هوا تاريخ الطباعه

```
PayBillsQueue.ServeNextClient();
cout << "\nPay Bills Queue After Se
PayBillsQueue.PrintInfo();

cout << "\nSubscriptions Queue Info

SubscriptionsQueue.IssueTicket();
```



```
Serve Time In
40 Minutes.

Pay Bills Queue After Serving One client

Queue Info

Prefix         = A0
Total Tickets  = 5
Served Clients = 1
```

```
cout << "\nSubscriptions Queue Info:\n";

SubscriptionsQueue.IssueTicket();
SubscriptionsQueue.IssueTicket();
SubscriptionsQueue.IssueTicket();
inline void clsQueueLine::IssueTicket()
Search Online

SubscriptionsQueue.PrintInfo();

SubscriptionsQueue.PrintTicketsLineRTL();
SubscriptionsQueue.PrintTicketsLineLTR();
```

اشتغل بال queue اللي موجود في ال stl مش اللي عملناه

Solution

ده الكلاس

```
#pragma once

#include <iostream>
#include "clsDate.h"
#include "queue"
#include "stack"

using namespace std;

class clsQueueLine
{

private:

    short _TotalTickets = 0;
    short _AverageServeTime = 0;
    string _Prefix = "";

    class clsTicket
    {
    private:

        short _Number = 0;
        string _Prefix;
        string _TicketTime;
        short _WaitingClients = 0;
        short _AverageServeTime = 0;
        short _ExpectedServeTime = 0;

    public:
        clsTicket(string Prefix, short Number, short WaitingClients, short
AverageServeTime)
        {

            _Number = Number;
            _TicketTime = clsDate::GetSystemDateTimeString();
            _Prefix = Prefix;
            _WaitingClients = WaitingClients;
            _AverageServeTime = AverageServeTime;
        }

        string Prefix()
        {
            return _Prefix;

        }
        short Number()
        {
            return _Number;
        }

        string FullNumber()
        {
            return _Prefix + to_string(_Number);
        }

        string TicketTime()
        {
            return _TicketTime;
        }

        short WaitingClients()
        {
```

```
            return _WaitingClients;
        }

        short ExpectedServeTime()
        {
            return _AverageServeTime * _WaitingClients;
        }

        void Print()
        {
            cout << "\n\t\t\t  _______________________\n";
            cout << "\n\t\t\t\t    " << FullNumber();
            cout << "\n\n\t\t\t    " << _TicketTime;
            cout << "\n\t\t\t    Wating Clients = " << _WaitingClients;
            cout << "\n\t\t\t      Serve Time In";
            cout << "\n\t\t\t       " << ExpectedServeTime() << " Minutes.";
            cout << "\n\t\t\t  _______________________\n";


        }
    };

public:

    queue <clsTicket> QueueLine;

    clsQueueLine(string Prefix, short AverageServeTime)
    {
        _Prefix = Prefix;
        _TotalTickets = 0;
        _AverageServeTime = AverageServeTime;
    }



    void IssueTicket()
    {
        _TotalTickets++;
        clsTicket Ticket(_Prefix, _TotalTickets, WaitingClients(), _AverageServeTime);
        QueueLine.push(Ticket);

    }

    int WaitingClients()
    {
        return QueueLine.size();

    }


    string WhoIsNext()
    {
        if (QueueLine.empty())
            return "No Clients Left.";
        else
            return QueueLine.front().FullNumber();

    }

    bool ServeNextClient()
    {
        if (QueueLine.empty())
            return false;


        QueueLine.pop();

        return true;

    }
```

```
    short ServedClients()
    {
        return _TotalTickets - WaitingClients();
    }

    void PrintInfo()
    {
        cout << "\n\t\t\t _________________________\n";
        cout << "\n\t\t\t\tQueue Info";
        cout << "\n\t\t\t _________________________\n";
        cout << "\n\t\t\t    Prefix    = " << _Prefix;
        cout << "\n\t\t\t    Total Tickets   = " << _TotalTickets;
        cout << "\n\t\t\t    Served Clients  = " << ServedClients();
        cout << "\n\t\t\t    Wating Clients  = " << WaitingClients(); ;
        cout << "\n\t\t\t _________________________\n";
        cout << "\n";

    }

    void PrintTicketsLineRTL()
    {

        if (QueueLine.empty())
            cout << "\n\t\tTickets: No Tickets.";
        else
            cout << "\n\t\tTickets: ";

        //we copy the queue in order not to lose the original
        queue <clsTicket> TempQueueLine = QueueLine;


        while (!TempQueueLine.empty())
        {
            clsTicket Ticket = TempQueueLine.front();

            cout << " " << Ticket.FullNumber() << " <-- ";

            TempQueueLine.pop();
        }

        cout << "\n";

    }

    void PrintTicketsLineLTR()
    {
        if (QueueLine.empty())
            cout << "\n\t\tTickets: No Tickets.";
        else
            cout << "\n\t\tTickets: ";

        //we copy the queue in order not to lose the original
        queue <clsTicket> TempQueueLine = QueueLine;
        stack <clsTicket> TempStackLine;

        while (!TempQueueLine.empty())
        {
            TempStackLine.push(TempQueueLine.front());
            TempQueueLine.pop();
        }

        while (!TempStackLine.empty())
        {
            clsTicket Ticket = TempStackLine.top();

            cout << " " << Ticket.FullNumber() << " --> ";

            TempStackLine.pop();
        }
        cout << "\n";
    }
```

```
    void PrintAllTickets()
    {

        cout << "\n\n\t\t\t       ---Tickets---";

        if (QueueLine.empty())
            cout << "\n\n\t\t\t     ---No Tickets---\n";

        //we copy the queue in order not to lose the original
        queue <clsTicket> TempQueueLine = QueueLine;


        while (!TempQueueLine.empty())
        {
            TempQueueLine.front().Print();
            TempQueueLine.pop();
        }

    }


};
```

وده ال main

```
//ProgrammingAdvices.com
//Mohammed Abu-Hadhoud

#include <iostream>
#include "clsQueueLine.h"

using namespace std;

int main()
{

    clsQueueLine PayBillsQueue("A0", 10);
    clsQueueLine SubscriptionsQueue("B0", 5);


    PayBillsQueue.IssueTicket();
    PayBillsQueue.IssueTicket();
    PayBillsQueue.IssueTicket();
    PayBillsQueue.IssueTicket();
    PayBillsQueue.IssueTicket();

    cout << "\nPay Bills Queue Info:\n";
    PayBillsQueue.PrintInfo();

    PayBillsQueue.PrintTicketsLineRTL();
    PayBillsQueue.PrintTicketsLineLTR();



    PayBillsQueue.PrintAllTickets();

    PayBillsQueue.ServeNextClient();
    cout << "\nPay Bills Queue After Serving One client\n";
    PayBillsQueue.PrintInfo();

    cout << "\nSubscriptions Queue Info:\n";

    SubscriptionsQueue.IssueTicket();
    SubscriptionsQueue.IssueTicket();
    SubscriptionsQueue.IssueTicket();


    SubscriptionsQueue.PrintInfo();
```

```
    SubscriptionsQueue.PrintTicketsLineRTL();
    SubscriptionsQueue.PrintTicketsLineLTR();

    SubscriptionsQueue.PrintAllTickets();

    SubscriptionsQueue.ServeNextClient();
    cout << "\nSubscriptions Queue After Serving One client\n";
    SubscriptionsQueue.PrintInfo();


    return 0;
}
```

The end