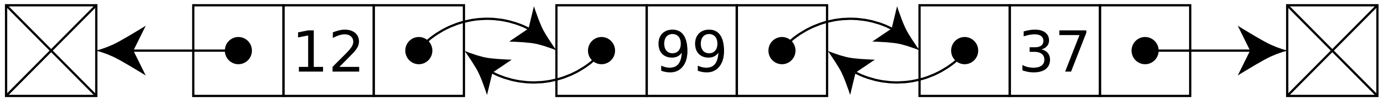
Lab 5

Doubly linked lists, classes, templates

Presentation by *Asem Alaa*

Doubly linked list



```
struct Node
{
    int data;
    Node *next;
    Node *prev;
};

struct DLL
{
    Node *front;
};

int main()
{
    DLL l{nullptr}; // l is empty linked list.
}
```

Doubly linked list

isEmpty?

```
bool isEmpty( DLL &list )
{
    return list.front == nullptr;
}
```

Doubly linked list

pushFront

```
void pushFront( DLL &list, int data )
{
    auto newNode = new Node{ data , list.front , nullptr };
    if( list.front ) list.front->prev = newNode;
    list.front = newNode;
}
```

Doubly linked list

pushBack

```cpp
void pushBack( DLL &list, int data )
{
    if( isEmpty( list ))
        pushFront( list, data );
    else
    {
        Node *temp = list.front;
        while( temp->next != nullptr )
            temp = temp->next;
        auto newNode = new Node{ data , nullptr , temp };
        temp->next = newNode;
    }
}
```

Doubly linked list

pushBack

```cpp
void pushBack( DLL &list, int data )
{
    if( isEmpty( list ))
        pushFront( list, data );
    else
    {
        Node *temp = list.front;
        while( temp->next != nullptr )
            temp = temp->next;
        auto newNode = new Node{ data , nullptr , temp };
        temp->next = newNode;
    }
}
```

Doubly linked list

popNode

```cpp
void popNode( DLL &list, Node *node )
{
    if( list.front == node ) list.front = list.front->next;
    if( node->prev ) node->prev->next = node->next;
    if( node->next ) node->next->prev = node->prev;
    delete node;
}
```

Doubly linked list

popFront

```
void popFront( DLL &list )
{
    popNode( list , list.front );
}
```

Doubly linked list

popBack

```
void popBack( DLL &list )
{
    if( isEmpty( list ))
        return;
    else if( list.front->next == nullptr )
        popFront( list );
    else
    {
        Node *temp = list.front;
        while( temp->next != nullptr )
            temp = temp->next;
        temp->prev->next = nullptr;
        delete temp;
    }
}
```

Doubly linked list

print

```cpp
void print( DLL &l )
{
    for( auto p = l.front; p != nullptr ; p = p->next )
    {
        std::cout << p->data << "->";
    }
    std::cout << "null\n";
}
```

Doubly linked list

size

```
int size( DLL &list )
{
    int counter = 0;
    auto temp = list.front;
    while( temp != nullptr )
    {
        ++counter;
        temp = temp->next;
    }
    return counter;
}
```

Doubly linked list

size

Or..

```cpp
int size( DLL &list )
{
    int counter = 0;
    for( auto temp = list.front; temp != nullptr; temp = temp->next )
        ++counter;
    return counter;
}
```

Doubly linked list

size (recursive)

```
int size( Node *front )
{
    if( front == nullptr ) return 0;
    else return 1 + size( front->next );
}
```