CPSC 2380-01 and 9S1: Algorithms

Project 1: Doubly Linked List

Due at the beginning of the class, Thursday, October 6, 2022, Thursday

By
Dr. Chia-Chu Chiang

Department of Computer Science
University of Arkansas at Little Rock
2801 S. University Avenue
Little Rock, Arkansas 7204-1099, USA

Total Points: 100 points

Given the code of doubly linked list,

// Node.h

```
#ifndef NODE_H
#define NODE_H

#include <iostream>
using namespace std;

class Node {
	friend class DoublyLinkedList;
	public:
		Node();
		Node(int v);
		~Node();


	private:
		Node *prev;
		int  value;
		Node *next;

};

#endif
```

// Node.cpp

```
#ifndef NODE_CPP
#define NODE_CPP

#include "Node.h"
Node::Node()
{
	prev = NULL;
	value = 0;
	next = NULL;
```

```
}

Node::Node(int v)
{
	prev = NULL;
	value = v;
	next = NULL;
}

Node::~Node()
{

}

#endif
```

// doublylinkedlist.h

```
#ifndef DOUBLYLINKEDLIST_H
#define DOUBLYLINKEDLIST_H
#include <iostream>
#include "Node.h"

class DoublyLinkedList{
public:
	DoublyLinkedList();
	~DoublyLinkedList();
	void addNewNodeToFront(Node* newNode);
	void addNewNodeToBack(Node* newnNde);
	Node* removeNodeFromFront();
	void removeNodeFromBack();
	void displayDoublyLinkedList();
	bool isPalindrome();
	void split(int n);
	void drawDoublyLinkedList();
private:
	Node* head;
	Node* tail;

};
#endif
```

// complete the following methods in the doubly linked list class.

// doublylinkedlist.cpp

```
#ifndef DOUBLYLINKEDLIST_CPP
#define DOUBLYLINKEDLIST_CPP

#include "doublylinkedlist.h"

DoublyLinkedList::DoublyLinkedList()
{
	head = NULL;
	tail = NULL;
}
```

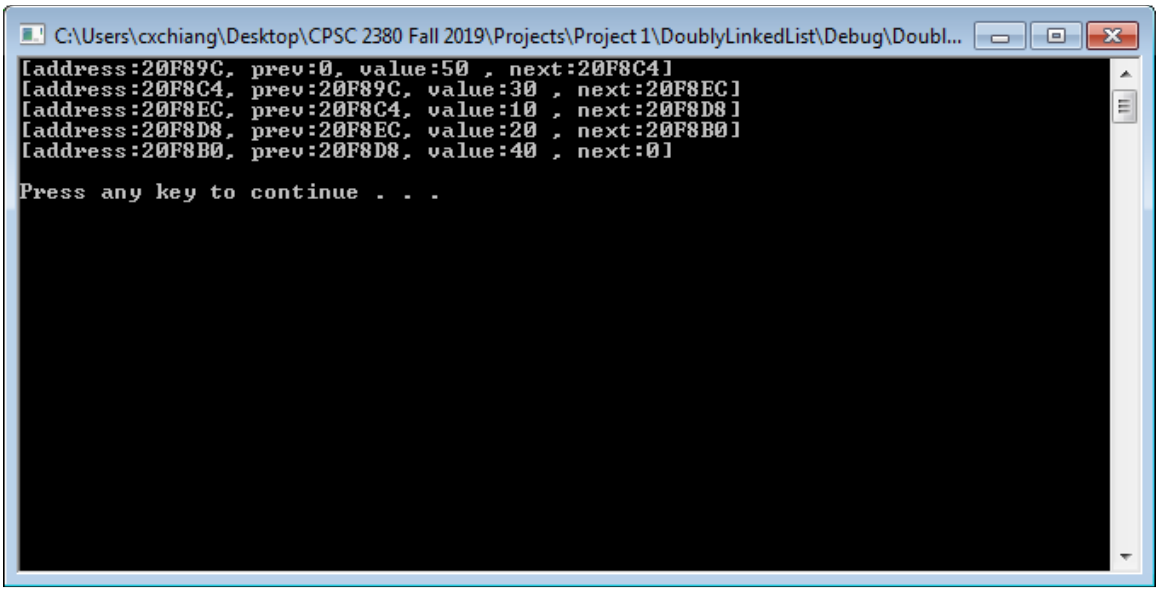
```
DoublyLinkedList::~DoublyLinkedList()
{

}

void DoublyLinkedList::addNewNodeToFront(Node * newNode) {
	if(head == NULL && tail == NULL) {
		head = tail = newNode;
	} else {
		head->prev = newNode;
		newNode->next = head;
		head = newNode;
	}
}

void DoublyLinkedList::addNewNodeToBack(Node * newNode) {
	if(head == NULL && tail == NULL) {
		head = tail = newNode;
	} else {
		tail->next = newNode;
		newNode->prev = tail;
		tail = newNode;
	}
}


Node* DoublyLinkedList::removeNodeFromFront()
{
	Node *tempNode;

	tempNode = head;
	head = head->next;
	return tempNode;
}


void DoublyLinkedList::displayDoublyLinkedList()
{
	Node *tempNode;

	tempNode = head;
	while (tempNode != NULL)
    {
        cout << tempNode->value << " ";
        tempNode = tempNode->next;
    }
}

void DoublyLinkedList::drawDoublyLinkedList()
{
  // CODE TO BE ADDED for Project 1
  // Might reuse displayDoublyLinkedList()
}
```

```
bool DoublyLinkedList::isPalindrome()
{
   // CODE TO BE ADDED for Project 1
   ...
}

Void DoublyLinkedList::split(int n) // n is the number of even partitions
{
   // CODE TO BE ADDED for Project 1
   ...
}
```

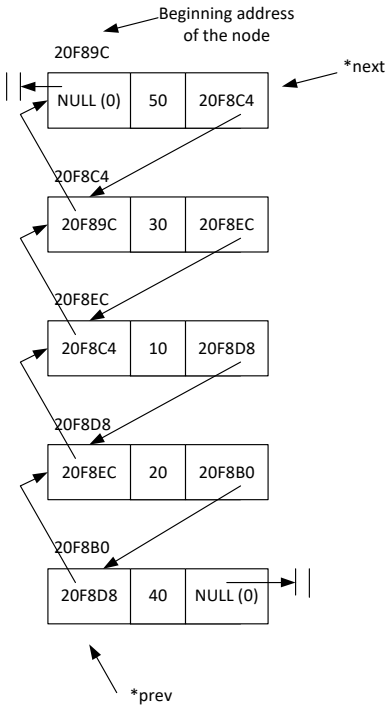
1. (30 Points) Write a method void drawDoublyLinkedList() that draws a pictorial view of a doubly linked list. For example, if you run the following main.cpp, you should produce the output below.

```
#include <iostream>
using namespace std;
#include "doublylinkedlist.h"

int main()
{
	// test case 1
	// Create a doubly linked list
	DoublyLinkedList Dll_1;

	// Each node contains only one integer of a string
	Node n11(10);
	Node n12(20);
	Node n13(30);
	Node n14(40);
	Node n15(50);

	// a doubly linked list of integers by inserting nodes
	Dll_1.addNewNodeToFront(&n11);
	Dll_1.addNewNodeToBack(&n12);
	Dll_1.addNewNodeToFront(&n13);
	Dll_1.addNewNodeToBack(&n14);
	Dll_1.addNewNodeToFront(&n15);

	Dll_1.drawDoublyLinkedList();

	cout << endl;


	system("PAUSE");
	return 0;
}
```

Output



The output shows the following of the pictorial picture of the doubly linked list.

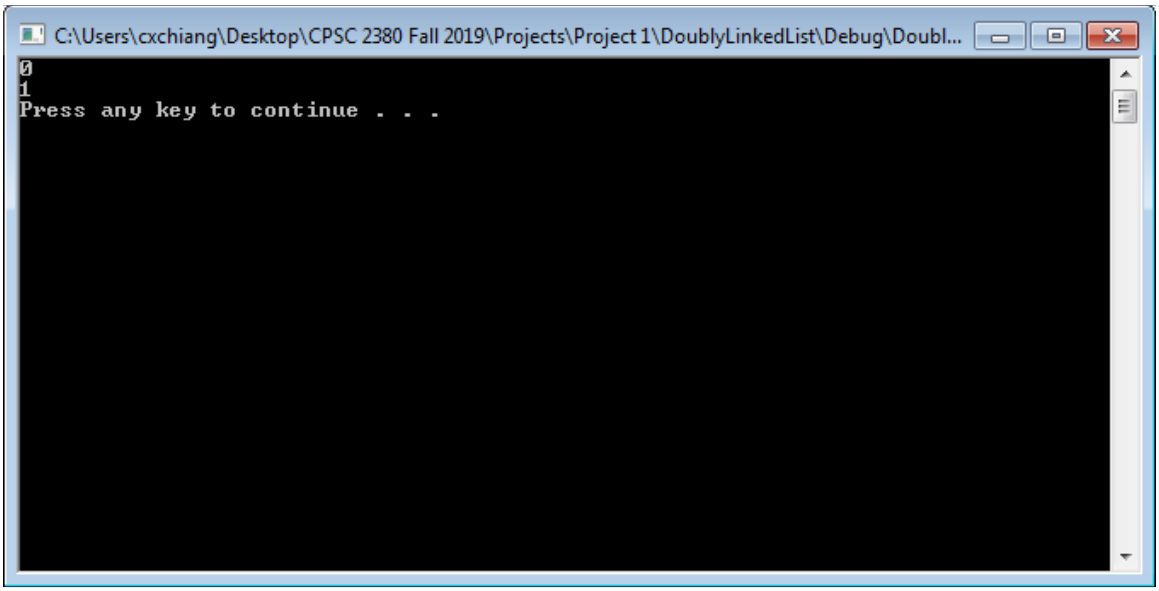


2. (30 Points) Write a method bool isPalindrome() that returns true if a doubly linked list of integers is a palindrome or not. An empty doubly linked list of integers is a palindrome. **You are NOT allowed to use any "reverse" method.** Please use the following main.cpp to test the method for correctness.

```
#include <iostream>
using namespace std;
#include "doublylinkedlist.h"
```

```
int main()
{
	// test case 1
	// Create a doubly linked list
	DoublyLinkedList Dll_1;

	// Each node contains only one integer of a string
	Node n11(10);
	Node n12(20);
	Node n13(30);
	Node n14(40);
	Node n15(50);

	// a doubly linked list of integers by inserting nodes
	Dll_1.addNewNodeToFront(&n11);
	Dll_1.addNewNodeToFront(&n12);
	Dll_1.addNewNodeToFront(&n13);
	Dll_1.addNewNodeToFront(&n14);
	Dll_1.addNewNodeToFront(&n15);

	cout << Dll_1.isPalindrome() << endl;


	// test case 2
	// Create a doubly linked list
	DoublyLinkedList Dll_2;

	// Each node contains only one integer of a string
	Node n21(10);
	Node n22(20);
	Node n23(20);
	Node n24(10);

	// a doubly linked list of integers by inserting nodes
	Dll_2.addNewNodeToFront(&n21);
	Dll_2.addNewNodeToFront(&n22);
	Dll_2.addNewNodeToFront(&n23);
	Dll_2.addNewNodeToFront(&n24);

	cout << Dll_2.isPalindrome() << endl;

	system("PAUSE");
	return 0;
}
```

Output

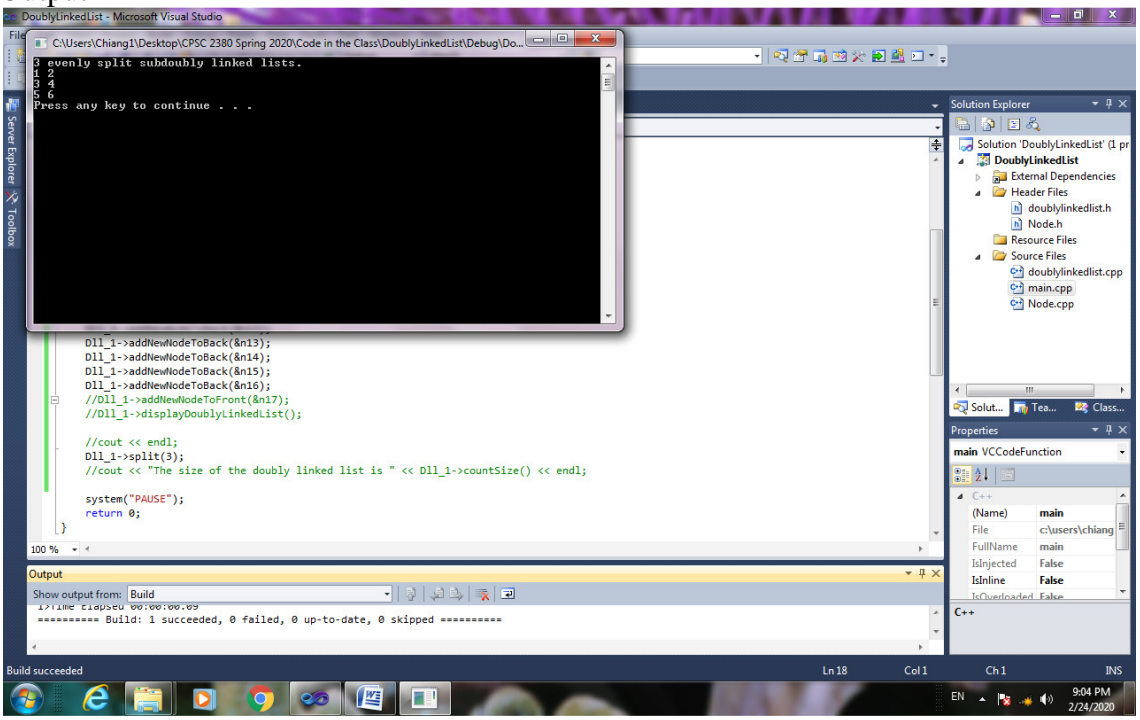


3. (40 Points) Write a method, void split(n), that **evenly** splits a doubly linked list into **n** (n ≥ 1) doubly sub linked lists. If n is less than 1, exceeds the size of the doubly linked list, or the list cannot be evenly divided, print "cannot be processed." to the output. Please use the following main.cpp to test the method for correctness.

```
// The main.cpp
#include <iostream>
using namespace std;

#include "Node.h"
#include "doublylinkedlist.h"

int main()
{
	// test case 1
	// Create a doubly linked list
	DoublyLinkedList *Dll_1;
	Dll_1 = new DoublyLinkedList;

	// Each node contains only one integer of a string
	Node n11(1);
	Node n12(2);
	Node n13(3);
	Node n14(4);
	Node n15(5);
	Node n16(6);

	// a doubly linked list of integers by inserting nodes
	Dll_1->addNewNodeToBack(&n11);
	Dll_1->addNewNodeToBack(&n12);
	Dll_1->addNewNodeToBack(&n13);
	Dll_1->addNewNodeToBack(&n14);
```

```
        Dll_1->addNewNodeToBack(&n15);
        Dll_1->addNewNodeToBack(&n16);

        Dll_1->split(3);

        system("PAUSE");
        return 0;
}
```

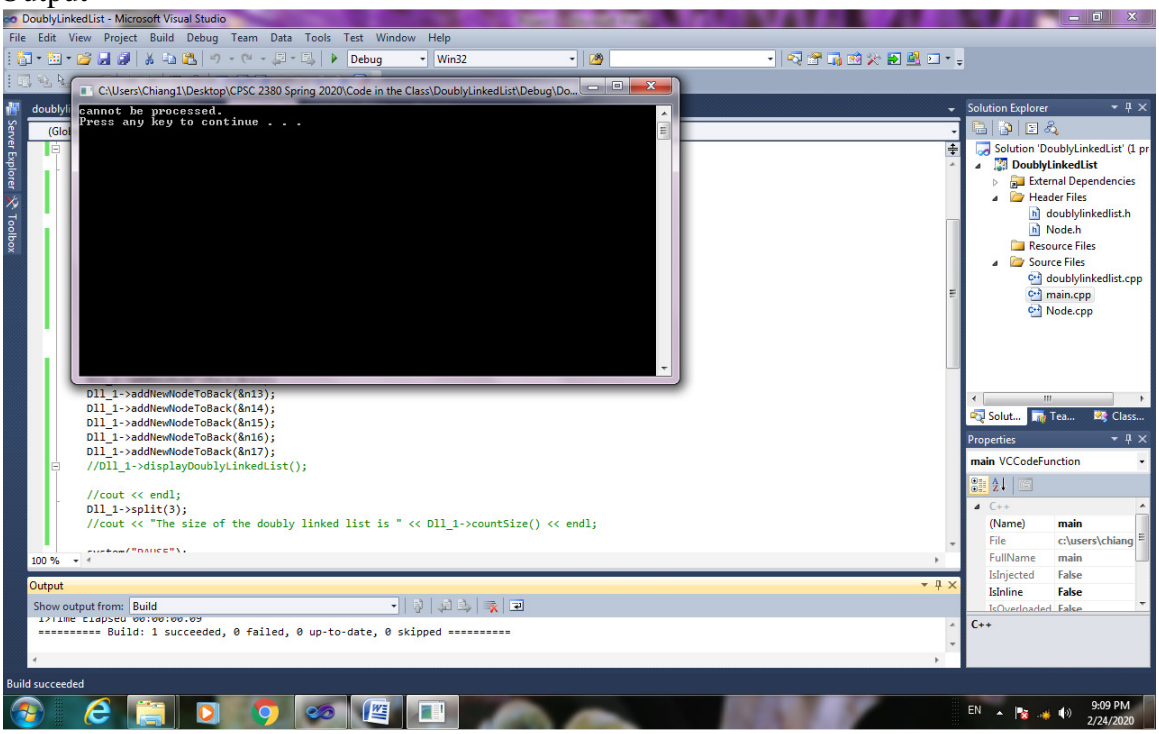
Output




```
// The main.cpp
#include <iostream>
using namespace std;

#include "Node.h"
#include "doublylinkedlist.h"

int main()
{
        // test case 2
        // Create a doubly linked list
        DoublyLinkedList *Dll_1;
        Dll_1 = new DoublyLinkedList;

        // Each node contains only one character of a string
        Node n11(1);
        Node n12(2);
        Node n13(3);
        Node n14(4);
        Node n15(5);
        Node n16(6);
        Node n17(7);
```

```
        // a doubly linked list of characters by inserting nodes
        Dll_1->addNewNodeToBack(&n11);
        Dll_1->addNewNodeToBack(&n12);
        Dll_1->addNewNodeToBack(&n13);
        Dll_1->addNewNodeToBack(&n14);
        Dll_1->addNewNodeToBack(&n15);
        Dll_1->addNewNodeToBack(&n16);
        Dll_1->addNewNodeToBack(&n17);

        Dll_1->split(3);

        system("PAUSE");
        return 0;
}
```

Output



Your project will be graded based on the correctness of the program. In addition, the readability of the program will also be taken into account. To make the program readable, you should have a program header placed on the top of the program describing your name, student ID, and a brief description of what the program is going to in one or two sentences. Your program should be modular and lots of comments in the program.

// University of Arkansas at Little Rock
// Department of Computer Science
// CPSC 2380-01 and 9S1: Algorithms
// Fall 2022
// Project 1: Doubly Linked List
// Due Date: October 6, 2022, Thursday

```
// Name:
// T-number (Last 4 Digits):
// Description of the Program (2-3 sentences):
// Date Written:
// Date Revised:
```

Figure 1: Program Header

Grading Scheme
1. No credits given if the program is not compiled.
2. Partial credits given if the program works partially.
3. Absolutely, no late project.

You should turn in the source code of project 1 to the Blackboard, with **sufficient** sample outputs. **LATE PROJECTS WILL NOT BE ACCEPTED. FIRM!**