Data Structures and Fundamentals of Programming (Questions 5-8)

Problem 5

In C++ implement a **generic** singly-linked-list class, called `List`, that uses **dynamic memory allocation**. It should be generic on the type of data stored in the list. This should implement the list ADT. The list should look something like the following:

front -> X_1 ->X_2 -> … -> X_n <- back

where X_1 is the first node in the list and X_n is the last node in the list. Besides `List`, you will most likely want another generic class or struct called `node`. Along with the class definition(s), you **must** implement the following methods for the generic `List` class:

- Default constructor
- Destructor
- Copy-constructor
- `AddToBack` which takes a parameter of type item and creates a new node that is added to the front of the list.
- `RemoveFromFront` which removes a node from the end of a list and returns its contents.

Note: Your implementation can **NOT** use STL or any other libraries (standard or otherwise).

Problem 6

Implement a function, in C++ to evaluate a **fully** parenthesized infix expression. You can assume the expression is correct. The infix expression will be passed into the function as a character array (null terminating) or string. The binary operators +, -, *, / with standard precedence are to be supported. You do not need to support unary operators. Additionally, you can assume that a generic class `Stack<Item>` exists with `push` and `pop` defined as normal and you may also use the C++ string class.

```
char expr1[] = "(2*((3+7)-10))";
string expr2 = "(16*((4+23)-7))";
```

Problem 7

Write a method that concatenates two strings (as defined below) and returns the resulting new string.

```
class string
{
  public:
    string();
    string operator+(string) const;

  private:
    char s[256];  //null terminated character array
};
```

Problem 8

A) Convert the following infix expressions into postfix and prefix.

```
a * b - c * d * e * (d – f) - g
```

```
a * (b + c) * (d - e) – d * f
```

B) Give the Preorder, Postorder, and Inorder traversals of the tree below: