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

Problem 5

In C++ implement a generic **double**-linked-list class, called `List<item>`, that uses dynamic memory allocation. The list must look like the following:

$$\texttt{frontptr -> } X_1 \texttt{ <->} X_2 \texttt{ <-> … <-> } X_n \texttt{ <- backptr}$$

where X_1 is the first node in the list and X_n is the last node in the list. Besides `List`, you will need class or struct called `dnode`. Along with the class definition(s), you must implement the following member functions for `List<item>`:

- `List()` - Default constructor
- `~List()` - Destructor
- `List(const List<item>&)` - Copy-constructor
- `AddBack(item)` – Adds an item to the back of the list.
- `AddFront(item)` – Adds an item to the front of the list.
- `item RemoveFront` – removes a node from the front of a list and returns its contents.

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

Problem 6

Write an iterator class called `ListItr` for the `List` class in the previous problem. For the iterator you must write the following member functions with standard meanings:

- `ListItr()` - Default constructor
- `ListItr(List<item>* ptr)` – Another constructor
- `operator++()` A pre-increment
- `operator++(int)` A post-increment
- `operator*()` – dereference

Additionally, for the `List` class write the following member functions:

- `front()` – returns an iterator to the front of the list
- `back()` – returns an iterator to the back of the list

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

Problem 7

Implement a function, in C++, to convert a fully parenthesized infix expression into a postfix expression. The binary operators +, -, *, / with standard precedence are to be supported. You do not need to support unary operators. You can assume that operands are the single letters (A-Z). The infix expression can be stored as a simple null terminating char string such as below or as a C++ string. 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 infix1[] = "(A*((C+D)-F))";
string infix2 = "(A*((C+D)-F))";
```

Problem 8

Implement an algorithm which, given a list of integers and a given number, determines whether a subset of the integers adds up to the number given. The program should output either the list which adds up to the number or an empty list in case of failure. You can assume that a list class has already been written.