Part A: In class we have built an abstract data type using an opaque object called MY_VECTOR. I would simply like for you to use the MY_VECTOR type to write a program what will open a file called numbers.txt containing integers separated by new line characters, read all of the numbers in the file into your vector and then print them back to a file called reverse_numbers.txt in the reverse order from the way they were found separated by new line characters. For example, if the file contains the numbers 41 7 33 100 5 then your program would output the file 5 100 33 7 41. Be sure to think about the benefit your vector provides to you even if the file were to contain a million numbers.

Part B: copy your MY_VECTOR type to two new files, char_vector.h and char_vector.c and modify it so that it can store characters instead of integers renaming functions as needed. Now use your new CHAR_VECTOR type to solve the parenthesis checker problem given below by writing a new main program:

You are given a string consisting of parenthesis style characters ( ) [ ] and { }. A string of this type is said to be correct if:

(a) It is an empty string.
(b) If string A is correct and string B is correct then string AB is correct.
(c) If A is correct then (A), [A] and {A} are all correct.

You are to write a program that takes as input from the keyboard a series of strings of this type that can be of any length. The first line of input will contain a single integer **n** telling you how many strings you will be testing. Following that integer will be **n** lines (each ending in a new line character) that represent the strings you are supposed to test with one string per line.

Your program should output **Yes** if a given string is correct and **No** otherwise.

For example if the user types the following as input

```
3
([])
(([{}]))
([()[]()])()
```

Then your program should give the following output:

```
Yes
No
Yes
```

How do you solve it? The trick to this problem is realizing that your vector type can behave as a stack data structure. One way to approach the problem is to read in a character, if it is a left marker then put it in the vector. If it is a right marker then check the last thing in the vector, if it is the correct left marker

then pop it out and continue testing the string. If you encounter a right marker and the last thing in the vector is not the left hand version of the marker you just picked up or the vector is empty then you can determine that the string is not correct, clear the vector and go on to the next example. If you finish a whole string (encounter a new line) without encountering a problem then you can determine that the input string is correct. If you are feeling industrious you can actually submit your program to an online checker that will test it for a small set of input values by going to uva.onlinejudge.org, making an account, and submitting your solution as a solution to problem 673. Even though the problem I have given you is a bit harder than the problem given there, your solution to this problem should have no problem solving their version of problem 673 that they have on the site.