Homework #4 - Arrays and Vectors

Name:

1 Intro

In C++, there's no obvious replacement for the Python list. Probably the simplest thing C++ has that is like a list is the **array**. That said, there are two big differences between Python lists and C++ arrays:

- You must know the size of a C++ array before you use it
- Everything in the array must be the same type (no mixing ints and strings)

Here is some simple code to create an array of 10 integers in C++:

```
1
#include <iostream>
3
using namespace std;
5
int main(){
7
    int myArray[10]; // Make an array that can hold 10 numbers
9   int i;

11  // Fill array
    cout << "Enter 10 numbers: ";
13  for (i = 0; i < 10; i++) {
      cin >> myArray[i];
15  }

17  // Print out array
    for (i = 0; i < 10; i++) {
19    cout << myArray[i] << " ";
    }
21  cout << endl;

23  return 0;
}
```

fixedArray.cpp

We can get around the need to know in advance how big the array is by using C++'s **new** and **delete** operators. These let us ask the computer for more memory while the computer is running. See the example on the next page.

```cpp

#include <iostream>

using namespace std;

int main(){
    int arraySize, i;

    cout << "How big do you want your array?: ";
    cin >> arraySize; // Ask for array size

    // Allocate a new array
    int* array = new (nothrow) int[arraySize];

    // Read in elements
    for (i = 0; i < arraySize; i++){
        cin >> array[i];
    }

    // Print array out backwards
    for (i = 0; i < arraySize; i++){
        cout << array[i] << " ";
    }
    cout << endl;

    // Free memory and return
    delete[] array;
    return 0;
}
```

dynamicArray.cpp

Here, we ask the user how large the array needs to be, and then **allocate** space in memory to hold it.

2 Vectors

Using "new" and "delete" creates a lot of extra work for us! In the example above we had to create an integer pointer (int *) and then use new to ask the computer for memory. What happens if later on we need the array to grow again? Or we want to make a copy of it into a new place in computer memory? These situations will make our code ugly and difficult to read. Luckily someone has already thought of this and has written the "vector" library! This library takes care of asking for memory for us, and makes our lives much simpler. A simple program to read N numbers into a vector and print them back out would look like this (see next page):

```cpp
// This code demonstrates basic vector usage in C++

#include <iostream>
#include <vector>

using namespace std;

int main(){
    int arraySize, i, tmp;
    vector<int> array;

    cout << "How big is your array?: ";
    cin >> arraySize;

    for (i = 0; i < arraySize; i++){
        cin >> tmp;
        array.push_back(tmp); // Vectors grow dynamically
    }

    for (i = 0; i < array.size(); i++) {
        cout << array[i] << " ";
    }
    cout << endl;

    return 0;

}
```

vectorUsage.cpp

See how much easier that is? No worrying about the final size of the vector, and no worrying about asking for more memory: the vector handles it all!

3 Now you try

3.1 Arrays Challenge

Use your new array knowledge to solve the arrays problem on HackerRank

3.2 Vector Challenge

Write a program which lets the user enter as many numbers as they want. When the user enters 9999, the program should stop and print out all the numbers that the user has entered so far. Like this:

```
Enter a number: 5
Enter a number: 25
Enter a number: 36
Enter a number: -4
Enter a number: 22
Enter a number: 9999
Your vector contains:
5 25 36 -4 22
```