Week 13 solutions

Exercises

1. Write a template for finding the absolute value of an variable `a` of type `T` for which the inequality `a < -a` is defined. Write a short program to test your template with different numerical types (e.g., `int`, `float`, and `double`).

```
#include <iostream>
using namespace std;

template <class T> const T abs(const T &a) {
  return (a >= -a) ? a : -a;
}

int main(int argc, const char *argv[]) {

  int i1 = -4, i2 = 9;
  float f1 = -1.0, f2 = 4.0;
  double d1 = -1e3, d2 = 0.0;

  cout << abs(i1) << ", " << abs(i2) << endl;
  cout << abs(f1) << ", " << abs(f2) << endl;
  cout << abs(d1) << ", " << abs(d2) << endl;

  return 0;
}
```

2. Read a series of floating point numbers from a text file and use a `vector` object the store them. After reading all numbers, the program should calculate and output the sum of the numbers and the mean of the numbers.

   Modify your program so that it prints the size of the vector (using the method `size()`) and its capacity (using the method `capacity()`) after reading a number. What is the complexity of appending a number to the vector?

```
#include <fstream>
#include <iostream>
#include <vector>
using namespace std;

int main(int argc, const char *argv[]) {

  string filename;
  double val = 0;
  vector<double> v;
```

```
  // Prompt user to enter filename
  cout << "Enter filename: ";
  cin >> filename;

  // Open file for input and check for errors
  ifstream myfile;
  myfile.open(filename);
  if (myfile.fail()) {
    cerr << "Error: " << strerror(errno) << endl;
    exit(EXIT_FAILURE);
  }

  // Append double precision numbers to v
  while (myfile >> val) {
    v.push_back(val);
    cout << "Size / Capacity: "
         << v.size() << " / " << v.capacity() << endl;
  }

  // Close file
  myfile.close();

  cout << "The file contained " << v.size() << " numbers\n";

  double sum = 0.0;
  for (int i = 0; i < v.size(); i++)
    sum += v[i];
  cout << "The sum of the numbers is " << sum << endl;
  cout << "The mean of the numbers is " << sum/v.size() << endl;

  return EXIT_SUCCESS;
}
```

We observe the following when running the program: the capacity of the array is doubled every time the current capacity is reach. This means the the **amortized** time complexity of adding n numbers to the end of a vector is $O(n)$.

3. Write a program that prompts the user to enter a sequence of strings, and store all the strings using a `list` object. When the user is done entering words, prompt the user to enter a number m and remove all strings that are longer than m characters from the list. Print the remaining elements in the list.

   What is the complexity of removing all strings of length at least m?

```
#include <iostream>
#include <list>
#include <string>
using namespace std;
```

```
int main(int argc, const char *argv[]) {

  int m = 0;
  string str;
  list<string> L;

  // Prompt user to enter strings
  do {
    cout << "Enter a string ('q' to stop): ";
    getline(cin, str);
    if (str.compare("q"))
      L.push_back(str);
  } while (str.compare("q"));
  cout << "You entered " << L.size() << " strings.\n";

  // Prompt user to enter m
  cout << "Enter a number m: ";
  cin >> m;

  // Delete strings of length at least m
  list<string>::iterator it;
  for (it = L.begin(); it != L.end();) {
    if (it->size() >= m)
      it = L.erase(it);
    else
      it++;
  }

  // Print remaining strings
  for (it = L.begin(); it != L.end(); ++it) {
    cout << *it << endl;
  }

  return 0;
}
```

The complexity of removing strings of length at least m from the list is $O(n)$ (assuming that the list contains n strings).