# p6.22

Henry Hsu

October 19, 2011

## 1 Specification

Write a function

```
void bar_chart(vector<double> data)
```

that displays a bar chart of the values in data, using asterisks, like this:

```
**********************
****************************************
****************************
*************************
**************
```

You may assume that all values in data are positive. First figure out the maximum value in data. That value's bar should be drawn with 40 asterisks. Shorter bars should use proportionally fewer asterisks.

## 2 Analysis/Design

Since no instructions was given by specification regarding data source, decision was made to solicit user for input values. A value cap of 100 was chosen to more visually demonstrate the differences in bar lengths.

- Obtain up to 5 user input, with a limit of 100
- Fill a string vector with caption titles
- Check for the largest value from the user input
- Use the largest value to calculate bar lengths for the remaining values
- Display the user input as a bar chart, with associated captions per input

# 3 Implementation

Obtain data input from user in the form of up to 5 numerical values, no greater than 100.

Create and fill a string vector with arbitrarily chosen captions. Then check for the largest value from the user input. Once determined, use this largest value as baseline to calculate the bar length for the largest value. This largest bar will be arbitrarily displayed as 40 * characters; the remaining values will be of bars of proportionally lesser length.

Display the user inputs as bars of varying length, commensurate with the size of their value; also attach the pre-filled caption vector to these bars.

"p6_22.cpp" 3≡

⟨ *include files* 5c ⟩
⟨ *get input* 4a ⟩
⟨ *fill captions* 4b ⟩
⟨ *create bar data* 5a ⟩
⟨ *bar chart* 5b ⟩

```
int main()
{
        vector <double> data = get_input ();
        vector <string> captions = fill_captions ();

        /* The following performs very basic input validation */
        if (data.size() == 0)
        {
                cout << "Program requires at least one valid input to continue." << endl;
        }
        else if (data.size() > 5)
        {
                cout << "Maximum of only 5 data entry allowed." << endl;
        }
        else
        {
                /* This code block checks the vector for the element with the largest value */
                double max_value = data[0];
                for (int counter = 1; counter < data.size (); counter++)
                {
                        if (data [counter] > max_value)
                        {
                                max_value = data [counter];
                        }
                }

                /* The below function creates a vector of % that matches the data vector elements
                vector<double> bar_data = create_bar_data(data, max_value);

                /* The below function outputs the user input along with the appropriate bars */
                bar_chart (data, bar_data, captions);

        }


}
```

◇

⟨ *get input* 4a ⟩ ≡

```
vector <double> get_input()
{
        vector <double> data;
        bool continue_input = true;
        while (continue_input)
        {
                cout << "Please input up to 5 values no greater than 100, use any letter
                double input;
                cin >> input;                   // input validation
                if (cin.fail() || input > 100 || input < -100)
                {
                        continue_input = false;
                }
                else
                {
                        data.push_back(abs(input));     // writes input to vector element
                }
        }
        return data;                            // returns user input as a vector
}
```

◇

Fragment referenced in 3.

⟨ *fill captions* 4b ⟩ ≡

```
vector <string> fill_captions()
{
        vector <string> captions(5);
        captions[0] = "Egypt";
        captions[1] = "France";
        captions[2] = "Japan";
        captions[3] = "Uruguay";
        captions[4] = "Switzerland";
        return captions;
}
```

◇

Fragment referenced in 3.

⟨ *create bar data* 5a ⟩ ≡

```
vector <double> create_bar_data(vector <double> database, double max_value)
{
        vector <double> bar_data;               // creates matching vector
        for (int counter = 0; counter < database.size(); counter++)
        {
                bar_data.push_back (((database[counter] / max_value) * 100));
        }

        return bar_data;
}
```

◇

Fragment referenced in 3.

⟨ *bar chart* 5b ⟩ ≡

```
void bar_chart(vector <double> data, vector <double> bar_data, vector <string> captions)
{
        for (int counter = 0; counter < bar_data.size (); counter++)
        {
                cout << right << setw(15) << captions[counter] << " ";
                for (double bar_counter = 0; bar_counter < bar_data[counter]; bar_counte
                {
                        cout << "*";
                }
                cout << endl;
        }
}
```

◇

Fragment referenced in 3.

These are the include files needed for library function calls

⟨ *include files* 5c ⟩ ≡

```
#include <iostream>
#include <iomanip>
#include <cmath>
#include <vector>

using namespace std;
```

◇

Fragment referenced in 3.

# 4 Test

e.g. if the data generated was: 10, 20, 40 Expect: ********** (10) ********** (20) ********** (40)

e.g. if the data generated was: 5, 10, 20 Expect: ********** (5) ********** (10) ********** (20)