```
Script started on 2021-02-24 12:20:08-0600
m_sadaf1@ares:~$ pwd
/home/students/m_sadaf1
m_sadaf1@ares:~$ cat again.info
Name: Madiha Sadaf
Class: CSC122 W01

Lab: "Oops...shall we try again?"
Option: Protect your inputs from user stupidity.

Level: 2
Level: +1
Total Level: 3

Description:

This program takes the input of two strings, compare them
without destroying their contents, and outputs the result
based on the return values given in the code.

m_sadaf1@ares:~$ cat again.tpq
Thought Provoking Questions:

1) By passing the argument as a string. You need the
   string header file: #include <string>.

2) To pass a string to a function, you need to pass it as a
   const by reference. The string cannot be modified after.

3) To pass a list of values to a function, you need to pass
   a vector. No elements need to be changed.

4) The function should take the minimum for int's and
   double's. They are not changed.

5) Int, double, char. It represents the value entered by
   the user.

6) If the function being used is reasonable or not. It can
   just check to see if the input is valid within the given
   range.

7) By overloading.

8) By using #indef, #define, and #endif.

9) Add #include to access the library in the main file.
   This helps compile all files at once.

10) Two files. One source file and one header file.
    #include the header file.

m_sadaf1@ares:~$ cat again.cpp
#include <iostream>
#include <string>
#include "input_prot.h"
#include <vector>

using namespace std;

int main() //Driver Function
{
    //Initializing Prompts

    const string error_prompt = "Invalid input.";

    const string menu = "\nPick a choice: \n\n"
            "1: Long Input Protection \n"
            "2: Double Input Protection \n"
            "3: Char Input Protection \n"
            "4: Long List Input Protection \n"
            "5: Double List Input Protection \n"
            "6: Char List Input Protection \n"
            "7: Quit \n";

    const string sub_menu = "\nPick a parameter: \n\n"
            "1: No extra parameter: \n"
            "2: Maximum \n"
            "3: Minimum \n"
            "4: Maximum and Minimum \n";

    // Initializing Input Variables

    double double_input;
    long long_input;
    char char_input;

    // Initializing Choices

    vector<double>double_list = {42.1, 4.2, 2.3};

    vector<long>long_list = {42, 4, 2};

    vector<char> char_list = {'A','B', 'C'};

    //Menu

    cout << "\t\t\tWelcome to the Input Protection"
            "Driver \n";
    char choice;
    bool complete = false;

    do
    {
        cout << menu;
        cin >> choice;
        switch (choice)
        {
```

```
        case '1': //Long Input Protection
        {
            cout << "Your choice: Long Input Protection\n";
            cout << sub_menu;
            bool valid = false;
            while (!valid)
            {
                cin >> choice;
                switch (choice)
                {
                    case '1': //No extra parameter
                    {
                        string prompt = "Enter a long: ";
                        input_prot(long_input, prompt,
                                error_prompt);
                        cout << "Valid choice: "
                                << long_input << endl;
                        valid = true;
                    }
                        break;

                    case '2': // Maximum
                    {
                        string prompt = "Enter a long < 5: ";
                        input_prot(long_input, 5, prompt,
                                error_prompt, true);
                        cout << "Valid choice: "
                                << long_input << endl;
                        valid = true;
                    }
                        break;

                    case '3': //Minimum
                    {
                        string prompt = "Enter a long > 9: ";
                        input_prot(long_input, 9, prompt,
                                error_prompt, false);
                        cout << "Valid choice: "
                                << long_input << endl;
                        valid = true;
                    }
                        break;

                    case '4': //Maximum and Minimum
                    {
                        string prompt = "Enter a long between 5 and 80: ";
                        input_prot(long_input, 5, 80, prompt,
                                error_prompt);
                        cout << "Valid choice: "
                                << long_input << endl;
                        valid = true;
                    }
                        break;
```

```
                    default:
                    {
                        cout << error_prompt << endl;
                    }
                }
            }
        }

            break;

        case '2': //Double Input Protection
        {
            cout << "Your choice: Double Input Protection\n";
            cout << sub_menu;
            bool valid = false;
            do
            {
                cin >> choice;
                switch (choice)
                {
                    case '1': //No extra parameter
                    {
                        string prompt = "Enter a double: ";
                        input_prot(double_input, prompt,
                                error_prompt);
                        cout << "Valid choice: "
                                << long_input << endl;
                        valid = true;
                    }
                        break;

                    case '2': //Maximum
                    {
                        string prompt = "Enter a double < 42.1: ";
                        input_prot(double_input, 42.1, prompt,
                                error_prompt, true);
                        cout << "Valid choice: "
                                << long_input << endl;
                        valid = true;
                    }
                        break;

                    case '3': //Minimum
                    {
                        string prompt = "Enter a double > 4.2: ";
                        input_prot(double_input, 4.2, prompt,
                                error_prompt, false);
                        cout << "Valid choice: "
                                << long_input << endl;
                        valid = true;
                    }
                        break;

                    case '4': //Maximum and Minimum
                {
                    string prompt = "Enter a double "
                             "between -4.2 and 42.1: ";
                    input_prot(double_input, -4.2, 42.1, prompt,
                            error_prompt);
                    cout << "Valid choice: "
                            << long_input << endl;
                    valid = true;
                }
                    break;

                default:
                {
                    cout << error_prompt << endl;
                }
            }
        }while (!valid);
    }

        break;

    case '3': // Char Input Protection
    {
        cout << "Your choice: Char Input Protection\n";
        string prompt = "Enter a char: ";
        input_prot(char_input, prompt, error_prompt);
        cout << "Valid choice: " << char_input << endl;
    }
        break;

    case '4': //Long List Input Protection
    {
        cout << "Your choice: Long List Input Protection\n";
        string prompt = "Pick 42, 4, or 2: ";
        input_prot(long_input, long_list,
                prompt, error_prompt);
        cout << "Valid choice: " << char_input << endl;
    }
        break;

     case '5': // Double List Input Protection
    {
        cout << "Your choice: Double List Input Protection\n";
        string prompt = "Pick 2.3, 4.2, or 42.1: ";
        input_prot(double_input, double_list,
                 prompt, error_prompt);
        cout << "Valid choice: " << double_input << endl;
    }
        break;

     case '6': // Char List Input Protection
    {
        cout << "Your choice: Char List Input Protection\n";
        string prompt = "Pick A, B, or C: ";
```

```
                input_prot(char_input, char_list,
                         prompt, error_prompt);
                cout << "Valid choice: " << char_input << endl;
            }
                break;

             case '7':
            {
               complete = true;
            }
                break;

            default:
            {
                if (!complete)
                {
                    cout << error_prompt;
                }
            }
        }
    }while(!complete);


    return 0;

}

m_sadaf1@ares:~$ cat input_prot.cpp
#include <iostream>
#include <string>
#include "input_prot.h"
#include <vector>
#include <cmath>
#include <limits>

using namespace std;

void input_prot(long & input, const string & prompt,
        const string & error_prompt)
{
    bool complete = false;
    while (complete == false)
    {
        cout << prompt;
        cin >> input;
        if (cin.peek() != '\n') //Any extra character
        {
            cerr << error_prompt << endl;
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::
                    max(), '\n');
            complete = false;
        }
        else if (cin.fail())
        {
            cerr << error_prompt << endl;
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::
                    max(), '\n');
            complete = false;
        }
        else
        {
            complete = true;
        }
    }
}

void input_prot(long & input, const long & bounce,
        const string & prompt, const string & error_prompt,
        const bool & bounce_max)
{
    bool complete = false;
    while (complete == false)
    {
        cout << prompt;
        cin >> input;
        if (cin.peek() != '\n')
        {
            cerr << error_prompt << endl;
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::
                    max(), '\n');
            complete = false;
        }
        else if (cin.fail())
        {
            cerr << error_prompt << endl;
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::
                    max(), '\n');
                    complete = false;
        }
        else if (!bounce_max) // Value is min
        {
            if (input < bounce)
            {
                cerr << error_prompt << endl;
                cin.clear();
                cin.ignore(numeric_limits<streamsize>::
                    max(), '\n');
                complete = false;
            }
            else
            {
                complete = true;
            }
        }
        else // Value is max
        {
            if (input > bounce)
            {
                cerr << error_prompt << endl;
                cin.clear();
                cin.ignore(numeric_limits<streamsize>::
                    max(), '\n');
                complete = false;
            }
            else
            {
                complete = true;
            }
        }
    }
}

void input_prot(long & input, const long & min,
        const long & max, const string & prompt,
        const string & error_prompt)
{
    bool complete = false;
    while (complete == false)
    {
        cout << prompt;
        cin >> input;
        if (cin.peek() != '\n')
        {
            cerr << error_prompt << endl;
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::
                    max(), '\n');
            complete = false;
        }
        else if (cin.fail())
        {
            cerr << error_prompt << endl;
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::
                    max(), '\n');
                    complete = false;
        }
        else if (input < min || input > max)
        {
            cerr << error_prompt << endl;
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::
                    max(), '\n');
            complete = false;
        }
        else
        {
            complete = true;
        }
    }
}

void input_prot(long & input, const vector<long> & list,
        const string & prompt, const string & error_prompt)
{
    bool complete = false;
    while (!complete)
    {
        input_prot(input, prompt, error_prompt);
        for (vector<long>::size_type pos = 0; pos <
                list.size(); pos++)
        {
            if (input == list[pos]) // If found in list, search is over
            {
                complete = true;
            }
        }
        if (!complete)
        {
            cerr << error_prompt << endl;
        }
    }
}

void input_prot(double & input, const string & prompt,
        const string & error_prompt)
{
    bool complete = false;
    while (complete == false)
    {
        cout << prompt;
        cin >> input;
        if (cin.peek() != '\n')
        {
            cerr << error_prompt << endl;
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::
                    max(), '\n');
            complete = false;
        }
        else if (cin.fail())
        {
            cerr << error_prompt << endl;
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::
                    max(), '\n');
            complete = false;
        }
        else
        {
            complete = true;
        }
    }
}

void input_prot(double & input, const double & bounce,
        const string & prompt, const string & error_prompt,
        const bool & bounce_max)
{
    bool complete = false;
    while (complete == false)
    {
        cout << prompt;
        cin >> input;
        if (cin.peek() != '\n')
        {
            cerr << error_prompt << endl;
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::
                    max(), '\n');
            complete = false;
        }
        else if (cin.fail())
        {
            cerr << error_prompt << endl;
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::
                    max(), '\n');
            complete = false;
        }
        else if (!bounce_max)
        {
            if(input < bounce)
            {
                cerr << error_prompt << endl;
                cin.clear();
                cin.ignore(numeric_limits<streamsize>::
                    max(), '\n');
                complete = false;
            }
            else
            {
                complete = true;
            }
        }
        else  //Value is max
        {
            if (input > bounce)
            {
                cerr << error_prompt << endl;
                cin.clear();
                cin.ignore(numeric_limits<streamsize>::
                    max(), '\n');
                complete = false;
            }
            else
            {
                complete = true;
            }
        }
    }
}

void input_prot(double & input, const double & min,
        const double max, const string & prompt,
        const string & error_prompt)
{
    bool complete = false;
    while (complete == false)
    {
        cout << prompt;
        cin >> input;
        if (cin.peek() != '\n')
        {
            cerr << error_prompt << endl;
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::
                    max(), '\n');
            complete = false;
        }
        else if (cin.fail())
        {
            cerr << error_prompt << endl;
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::
                    max(), '\n');
            complete = false;
        }
        else if (input < min || input > max)
        {
            cerr << error_prompt << endl;
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::
                    max(), '\n');
            complete = false;
        }
        else
        {
            complete = true;
        }
    }
}

void input_prot(double & input, const vector<double> & list,
        const string & prompt, const string & error_prompt)
{
    bool complete = false;
    while (complete == false)
    {
        input_prot(input, prompt, error_prompt);
```

```
        for (vector<double>::size_type pos = 0; pos <
                list.size(); pos++)
        {
            if (fabs((input - list[pos])) < 1e-6) // If found in list, search is o
            {
                complete = true;
            }
        }
        if (!complete)
        {
            cerr << error_prompt << endl;
        }
    }
}

void input_prot(char & input, const string & prompt,
        const string & error_prompt)
{
    bool complete = false;
    while(!complete)
    {
        cout << prompt;
        cin >> input;
        if (cin.peek() != '\n')
        {
            cerr << error_prompt << endl;
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::
                    max(), '\n');
            complete = false;
        }
        else
        {
            complete = true;
        }
    }
}

void input_prot(char & input, const vector<char> & list,
        const string & prompt, const string & error_prompt)
{
    bool complete = false;
    while(complete == false)
    {
        input_prot(input, prompt, error_prompt);
        for (vector<char>::size_type pos = 0;
                pos < list.size(); pos++)
        {
            if (input == list[pos])
            {
                complete = true;
            }
        }
        if (!complete)
        {
            cerr << error_prompt << endl;
        }
    }
}

// Adds string to vector

void input_char(const string & input,
        vector<char> & list)
{
    for (string::size_type pos = 0; pos < input.length();
            pos++)
    {
        list.push_back(input[pos]);
    }
}

m_sadaf1@ares:~$ cat input_prot.h
#ifndef INPUT_H
#define INPUT_H

#include <string>
#include <cstdlib>
#include <vector>

// Checks to see if input is long

void input_prot(long & input, const std::string & prompt,
const std::string & error_prompt);

// Max or Min

void input_prot(long & input, const long & bounce,
        const std::string & prompt, const std::string & error_prompt,
        const bool & bounce_max);

//Max and Min

void input_prot(long & input, const long & min,
        const long & max, const std::string & prompt,
        const std::string & error_prompt);

// Checks to see if input is in list of longs

void input_prot(long & input, const std::vector<long>
        & list, const std::string & prompt,
        const std:: string & error_prompt);

// Checks to see if input is double

void input_prot(double & input, const std::string & prompt,
        const std::string & error_prompt);

// Max or Min

void input_prot(double & input, const double & bounce,
        const std::string & prompt, const std::string &
        error_prompt, const bool & bounce_max);

// Max and Min

void input_prot(double & input, const double & min, const
        double max, const std:: string & prompt,
        const std::string & error_prompt);

// Checks to see if input is in list of doubles

void input_prot(double & input, const
        std::vector<double> & list_input,
        const std::string & prompt,
        const std::string & error_prompt);

// Checks to see if input is char

void input_prot(char & input, const std::string & prompt,
        const std:: string & error_prompt);

// Checks to see if input is in list of chars

void input_prot(char & input, const std:: vector<char>
        & list, const std::string & prompt,
        const std:: string & error_prompt);

// Adds string to vector

void input_char(const std:: string & input,
        std::vector<char> & list);

#endif /*INPUT_H*/

m_sadaf1@ares:~$ CPP again input_prot
again.cpp***
input_prot.cpp...


m_sadaf1@ares:~$ ./again.out
                        Welcome to the Input ProtectionDriver

Pick a choice:

1: Long Input Protection
2: Double Input Protection
3: Char Input Protection
4: Long List Input Protection
5: Double List Input Protection
6: Char List Input Protection
7: Quit
```

```
1
Your choice: Long Input Protection

Pick a parameter:

1: No extra parameter:
2: Maximum
3: Minimum
4: Maximum and Minimum
1
Enter a long: 2134
Valid choice: 2134

Pick a choice:

1: Long Input Protection
2: Double Input Protection
3: Char Input Protection
4: Long List Input Protection
5: Double List Input Protection
6: Char List Input Protection
7: Quit
2
Your choice: Double Input Protection

Pick a parameter:

1: No extra parameter:
2: Maximum
3: Minimum
4: Maximum and Minimum
2
Enter a double < 42.1: 42.0
Valid choice: 2134

Pick a choice:

1: Long Input Protection
2: Double Input Protection
3: Char Input Protection
4: Long List Input Protection
5: Double List Input Protection
6: Char List Input Protection
7: Quit
3
Your choice: Char Input Protection
Enter a char: S
Valid choice: S

Pick a choice:

1: Long Input Protection
2: Double Input Protection
3: Char Input Protection
```

```
4: Long List Input Protection
5: Double List Input Protection
6: Char List Input Protection
7: Quit
4
Your choice: Long List Input Protection
Pick 42, 4, or 2: 2
Valid choice: S

Pick a choice:

1: Long Input Protection
2: Double Input Protection
3: Char Input Protection
4: Long List Input Protection
5: Double List Input Protection
6: Char List Input Protection
7: Quit
5
Your choice: Double List Input Protection
Pick 2.3, 4.2, or 42.1: 4.2
Valid choice: 4.2

Pick a choice:

1: Long Input Protection
2: Double Input Protection
3: Char Input Protection
4: Long List Input Protection
5: Double List Input Protection
6: Char List Input Protection
7: Quit
6
Your choice: Char List Input Protection
Pick A, B, or C: B
Valid choice: B

Pick a choice:

1: Long Input Protection
2: Double Input Protection
3: Char Input Protection
4: Long List Input Protection
5: Double List Input Protection
6: Char List Input Protection
7: Quit
8
Invalid input.
Pick a choice:

1: Long Input Protection
2: Double Input Protection
3: Char Input Protection
4: Long List Input Protection
```

```
5: Double List Input Protection
6: Char List Input Protection
7: Quit
7
m_sadaf1@ares:~$ exit
exit

Script done on 2021-02-24 12:22:23-0600
```