Name: Otto Wiking Häger

Date: 2/19/24

CSCI 1300: Recitation 5

Spring 2024

Please make sure to write your name and the date in the top left corner. You may use any course materials to answer the following questions and you may collaborate with others, but the work shown must be your own.

1 Spot The Error

Problem 1.1. The program intends to prints all even numbers from 2 to N (both inclusive). Identify the error(s) in the code below, and write the correct line(s).

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

void printEvenNumbers(int N)
{
    int i = 2;
    while (i <= N)
    {
        if (i % 2 == 0)
        {
            cout << i << " ";
        }
    } ←—i++
    return;
}

int main()
{
    int number;
    cout << "Enter a number: " << endl;
    cin >> number;
    printEvenNumbers(number);
    return 0;
}
```

Problem 1.2. The program monitors the pump status and fills the liquid until the liquid level reaches the threshold. Identify the error(s) in the code below, and write the correct line(s).

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

int main()
{
    int liquid_level = 5;
    int threshold = 60;
                             ↙ <=
    while (liquid_level >= threshold)
    {
        cout << "Pump is running. Liquid level: " << liquid_level << " units." << endl;
        liquid_level += 5;
                      ↖ -=
    }

    cout << "Pump stopped. Liquid level: " << liquid_level << " units." << endl;

    return 0;
}
```

Name:

Date:

CSCI 1300: Recitation 5

Spring 2024

Problem 1.3. The program intends to verify if two strings match, prompting the user to re-type the string until it matches the initial string. Identify the error(s) in the code below, and write the correct line(s).

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

int main()
{
    string initial_string;
    string retype_string;

    cout << "Enter your string: ";
    cin >> initial_string;

    cout << "Enter your string again: ";
    cin >> retype_string;
                                  !=
    while (retype_string == initial_string)
    {
        cout << "Your strings do not match. Try again: " << endl;
        cin >> retype_string;
    }

    cout << "Your strings match!!!!" << endl;
    return 0;
}
```

Problem 1.4. The program intends to find the sum of all numbers from 1 to N (inclusive) and prints the result. Identify the error(s) in the code below, and write the correct line(s).

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

void totalSum(int n)
{
    int result = 1;
    for(i = 1; i <= n; i+1)   i=i+1 / i++
    {
        result += i;
    }
    return result;
}

int main()
{
    int n;
    cout << "Enter a positive integer (n): ";
    cin >> n;

    int result = totalSum(n);
    cout << "Sum of numbers from 1 to " << n << " is: " << result << endl;

    return 0;
}
```

Name:

Date:

CSCI 1300: Recitation 5

Spring 2024

Problem 1.5. The program intends to calculate the sum of numbers entered by the user until a negative number is encountered. Identify the error(s) in the code below, and write the correct line(s).

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

int main()
{
     int num; ← number
     int sum = 0;

     do
     {
          cout << "Enter a number (enter a negative number to stop): ";
          cin >> number;
          if (number >= 0)
          {
               sum += number;
          }

     }(while number >= 0);

     cout << "Sum of the entered numbers: " << sum << endl;
     return 0;
}
```

Name:

Date:

CSCI 1300: Recitation 5

Spring 2024

2 Valid Integer

Design a function `validateInt` that accepts a string input and determines if it represents a valid integer by checking if each character in the string is a valid value. Your program should ask the user to input an integer, store it as a string and then invoke the `validateInt` function to check its validity. The program should then print whether the string is a valid integer or not. (Negative integers are also valid integers).

Function: validateInt(string)	`bool validateInt(string input)`
Purpose:	Iterate through a string and verify if it is a valid integer or not.
Parameters:	**input** - The string to be verified
Return value:	It returns true is the string is a valid integer. Otherwise returns false.
Error handling/ Boundary conditions:	If length of input = 0, false is returned
Example:	

Sample Code 1:

```
// Assume the proper libraries are included.

// Assume the proper implementation of
↪  validateInt() is included.

int main()
{
    string number;
    cout << "Enter the integer : " << endl;
    getline(cin, number);
    if(!validateInt(number))
    {
        cout << "The entered string is not a
        ↪  valid integer!!" << endl;
    }
    else
    {
        cout << "The entered string is a valid
        ↪  integer!!" << endl;
    }
    return 0;
}
```

Sample Output 2.1:

```
Enter the integer :
1234
The entered string is a valid integer!!
```

Here are a few more sample runs:

Sample Output 2.2:

```
Enter the integer :
-12
The entered string is a valid integer!!
```

Sample Output 2.3:

```
Enter the integer :
23e56
The entered string is not a valid integer!!
```

Sample Output 2.4:

```
Enter the integer :
32 56
The entered string is not a valid integer!!
```

Sample Output 2.5:

```
Enter the integer :
-
The entered string is not a valid integer!!
```

Problem 2.1. Write out the steps you would use to solve this problem by hand as pseudocode.

```
Give instructions to user.
tak input.
Save in string.
call Validation function
if input[0]=='-'
delete '-'
iterate through string, characters.
if a 'char' is False when given to isdigit.
return false.
esle return true if loop is done.
```

Name:

Date:

CSCI 1300: Recitation 5

Spring 2024

Problem 2.2. Pick four possible inputs for your program. Try to pick values that will test different aspects of your function. Follow the steps you wrote for these values to find your result, and verify it.

-1 → int

a → not int

1a → not int

-736.7 → non int

Problem 2.3. Translate each of the sample inputs and expected outputs you created into assert statements.

```
assert(validate(-1) == true)
assert(validate(a) == false)
assert(!validate(1a))
assert(!validate(-736.7))
```

Problem 2.4. Implement your solution in C++ using VS Code. Revise your solution, save, compile and run it again. Are you getting the expected result and output? Keep revising until you do. Make you sure you test for the values used in your sample runs, and for the boundary conditions.

2 3 4 5
~~A B C D E~~
Name: 0 1 2 3 4 l < 5

Spring 2024

Date: CSCI 1300: Recitation 5

3 Fitness Tracker

You are tasked with developing an fitness tracking program that allows users to track their daily exercise routines. Users can enter the the type of exercise and the duration (in minutes), and they can continue entering exercise until they decide to stop. The program should provide a summary of exercise time (in minutes) by type and calculate the total exercise time (in minutes) for the day.

Your program should follow these guidelines:

1. Create 3 integer variables - Cardio, Strength, Flexibility - to store the category-wise totals.
2. Allow the user to enter exercise by providing category and time (in minutes). If they have already added time to that category, it should add the new value to the current total for that category.
3. Repeat this until the user decides to stop - which is done by inputting exit in the category prompt.
4. If the user inputs a category that doesn't exist, reprompt the user to input the right one.
5. Display the category-wise totals and calculate the total exercise time (in minutes) for the day.

Sample Output 3.1:

```
Enter a category (Cardio, Strength, Flexibility, or 'exit'):
Flexibility
Enter the exercise time:
30
Enter a category (Cardio, Strength, Flexibility, or 'exit'):
Strength
Enter the exercise time:
40
Enter a category (Cardio, Strength, Flexibility, or 'exit'):
Endurance
Invalid Category.  Please enter a valid category.
Enter a category (Cardio, Strength, Flexibility, or 'exit'):
Cardio
Enter the exercise time:
20
Enter a category (Cardio, Strength, Flexibility, or 'exit'):
exit
Category-wise totals:
Cardio:  20 mins
Strength:  40 mins
Flexibility:  30 mins
Total exercise time for the day:  90 mins
```

Problem 3.1. Write out the steps you would use to solve this problem by hand as pseudocode.

```
Get user input for exercise.
Collect time exercised.

if exit.
Print out all times
```

Name: Spring 2024

Date:

CSCI 1300: Recitation 5

Problem 3.2. Pick possible inputs for your program. Choose as many inputs as you think you need to thoroughly test your program. Follow the steps you wrote for these values to find your result, and verify it.

Pilates → Enter valid category

cardio → enter Valid category

Cardio → enter time trained

" " → enter valid category.

Problem 3.3. Translate your pseudocode into a c++ program to solve the above code, using your assert statements in your main function to verify that your program works as expected.

Submission Instructions: Create a zip file that contains your solution .cpp file for question 2, question 3, and photos of this handout and submit on Canvas.