Assignment 1

- ***Condition and Loop***

Due: 20th October, Sunday, 23:59

1. This assignment contains ***TWO questions***. You are required to ***submit*** a complete C++ program (.cpp only) for each question on ***PASS*** ***before the deadline (submission will still be allowed one week after the deadline, but a score penalty of 50% will be imposed)*****.**

2. You can submit as many times as you want before the deadline, and we will ***grade your latest version***.

3. ***Only a small set of the test cases are visible for the testing, and a more comprehensive set of test cases will be used for grading.*** In other word, passing all the visible test cases does not mean that you can get full mark for this assignment. Try your best to thoroughly test your program. ***Hidden test cases will not be released.***

4. The marking of each question is based on the percentage of the total test cases that your solution can pass. ***If your submitted solution leads to a compilation error on PASS, zero mark will be given to that question and no manual checking to your solution is provided in such a case.***

5. ***No late submission*** will be accepted. ***ALL*** submissions should be on ***PASS***.

6. ***Plagiarism check*** will be performed.

7. You ***only need to use*** the material from ***Lectures 1 to 4***. It is NOT necessary to include any other library, except <iostream> and <limits>.

Question 1 [3 mark] ***Find the Smallest Number***

Write a simple program that helps a user find the smallest positive number among a limited series of positive integers entered by the user. The program should perform the following tasks:

1. **Prompt the user to enter a series of positive integers:** The program should repeatedly prompt the user to enter up to 5 positive integers, separated by blanks. The user can enter no more than 5 positive integers, and to indicate the end of input, the user should enter -1. The program will omit the integers after -1.

    - **Wrong Condition 1:** If the user enters a non-positive integer (excluding -1), display an error message and terminate the program. The program should not consider negative integers.

        – Example: -2 3 4 1 -1 *Note: -2 is a non-positive integer before ending*

        – Example: 0 3 4 1 -1 *Note: 0 is a non-positive integer before ending*

        – Example: 3 -4 -1 *Note: -4 is a non-positive integer before ending*

        – Example: 3 2 -4 -1 *Note: -4 is a non-positive integer before ending*

        – Example: 3 2 -4 9 -1 *Note: -4 is a non-positive integer before ending*

        For these examples, the program display:

        ```
        Error: Non-positive integer.

        Thank you for using the program!
        ```

    - **Wrong Condition 2:** If the user enters a non-integer value (e.g., a letter or a symbol), display an error message and terminate the program. The program should not consider this input.

        – Example: 5 B C 2 -1 *Note: B is a non-integer value before ending*

        – Example: B 2 -1 *Note: B is a non-integer value before ending*

        – Example: $ 1 4 -1 *Note: $ is a non-integer value before ending*

        – Example: A 5 B C 2 -1 *Note: A is a non-integer value before ending*

        – Example: 5 2 4 1 & -1 *Note: & is a non-integer value before ending*

        For these examples, the program display:

```
Error: Non-integer value.
Thank you for using the program!
```

- **Wrong Condition 3:** If the user only enters -1 or the first integer is -1, display an error message and terminate the program.

  – Example: -1 *Note: only enters -1*

  – Example: -1 2 5 3 *Note: the first integer is -1*

  For these two examples, the program display:

```
Error: Only -1.
Thank you for using the program!
```

- **Wrong Condition 4:** If the number of positive numbers is more than 5 before the ending -1, display an error message and terminate the program.

  – Example:6 5 4 3 2 1 -1 *Note: the number of positive numbers is more than 5 before the ending -1.*

  – Example: 3 4 2 5 3 4 2 3 -1 *Note: the number of positive numbers is more than 5 before the ending -1.*

  For these two examples, the program display:

```
Error: Not ended with -1.
Thank you for using the program!
```

The program will only report the first error it meets.

– Example: -2 A 15 20 25 -1 *Note: the first error is non-positive integer.*

For this example, the program display:

```
Error: Non-positive integer.
Thank you for using the program!
```

2. **Find the smallest positive number:** After the user finishes entering feasible numbers and enters -1 to stop, the program should determine the smallest positive number from the entered values.
3. **Display the smallest positive number:** The program should display the smallest positive number to the user.
4. **Ask the user if they want to repeat the process:** After displaying the smallest positive number, the program should ask the user if they want to find the smallest positive number again with a new series of positive integers. If the user responds with "Y", repeat the process. If the user responds with "N", end the program. Other responses, such "abc" and "aaa", will also end the program (*Please refer to the Example 5 below*).

The examples have considered most cases for test. Please carefully read the examples, which will help you to fully understand the above question.

Ensure that the program output follows the format of the following examples. Otherwise, you will not pass the test.

For example, "Enter up to 5 positive integers, separated by blanks. To stop, enter -1 and press Enter:". You should not misspell any words. A good strategy is to directly copy the sentence and use it in your program.

Example 1

```
Enter up to 5 positive integers, separated by blanks. To stop, enter -1 and press Enter:

10 15 20 -1

The smallest positive number is: 10

Do you want to find the smallest positive number again? (Y/N):

Y

Enter up to 5 positive integers, separated by blanks. To stop, enter -1 and press Enter:

5 8 12 18 22 -1

The smallest positive number is: 5

Do you want to find the smallest positive number again? (Y/N):

N

Thank you for using the program!
```

Example 2

```
Enter up to 5 positive integers, separated by blanks. To stop, enter -1 and press Enter:

-2 3 -1

Error: Non-positive integer.

Thank you for using the program!
```

Example 3

```
Enter up to 5 positive integers, separated by blanks. To stop, enter -1 and press Enter:

5 B C D E -1

Error: Non-integer value.

Thank you for using the program!
```

Example 4

```
Enter up to 5 positive integers, separated by blanks. To stop, enter -1 and press Enter:

-2 A 15 20 25 -1

Error: Non-positive integer.

Thank you for using the program!
```

Example 5

```
Enter up to 5 positive integers, separated by blanks. To stop, enter -1 and press Enter:

17 -1

The smallest positive number is: 17

Do you want to find the smallest positive number again? (Y/N):

aaa

Thank you for using the program!
```

Example 6

```
Enter up to 5 positive integers, separated by blanks. To stop, enter -1 and press Enter:

-1

Error: Only -1.

Thank you for using the program!
```

Example 7

```
Enter up to 5 positive integers, separated by blanks. To stop, enter -1 and press Enter:
6 5 4 3 2 1 -1

Error: Not ended with -1.

Thank you for using the program!
```

Example 8

```
Enter up to 5 positive integers, separated by blanks. To stop, enter -1 and press Enter:

2 2 2 -1

The smallest positive number is: 2

Do you want to find the smallest positive number again? (Y/N):

N

Thank you for using the program!
```

Hint:

To check wrong condition-2, you can use the following code.

```
#include <limits>
int userInput;
bool wrongCondition2 = false;
cin >> userInput;

// Check for wrong condition 2: Non-integer value
if (cin.fail()) {
    cin.clear();
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
    cout << "Error: Non-integer value.\n";
    wrongCondition2 = true;
    break; // Terminate the program after reporting the error.
}
```

If your extra input is unexpectedly used in the next iteration, you can use the following code to clear the extra input.

```
// Clear the input buffer
cin.clear();
cin.ignore(numeric_limits<streamsize>::max(),   '\n');
```

Question 2 [3 mark] ***Birthday Gift***

One day, Mike wanted to buy a birthday gift for his friend Judy. At that time, his wallet contains four kinds of banknotes (i.e., banknotes with denominations of ***1$, 3$, 5$, and 10$***), and the number of each banknote is ***a, b, c, and d***, respectively. If he needs to pay ***n*** dollars to buy the birthday gift, can he ***pay successfully without change making***? ***If the payment is successful (without change), please calculate the number of payment methods; if the payment is unsuccessful (with change), please output no.***

Write a program that reads a string of positive integers (i.e., ***a, b, c, d, n***, where data type: ***int***, ***with spaces between integers***) as input and outputs the number of successful payment methods or no.

[Example]

Suppose Mike has ***1$, 3$, 5$, and 10$ each (i.e., a=b=c=d=1)*** and he wants to buy a birthday gift worth ***15$ (n=15)***. As a result, he can pay successfully without making change, and he has ***only one*** payment method, namely ***one 5$ and one 10$.***

[Hint]

You can use a nested loop structure to enumerate all possible cases.

Sample Input and Output

The program reads the number of 1$, 3$, 5$, 10$ banknotes as well as the gift price sequentially, i.e., a, b, c, d, n. The program then outputs the number of successful payment methods or no.

Example 1

```
1 1 1 1 1

1
```

Example 2

```
1 2 3 4 15

2
```

Example 3

```
1 1 1 1 100

no
```

Example 4

```
5 2 3 4 15
```

```
7
```

Example 5

```
5 6 4 4 25

18
```