

SC1008 Lab 1 of C++ - Basic C++ Programming

1. **(Valid Input Data Type)** In Q2 and Q4 of Tutorial 1, we assume the users will always input the correct type of data we requested. But this is actually not true in real world. Write a C++ program to ask users to input student information, including name (`char []`), student ID (`int`) and math mark (`float`). You need to check if the user input the **correct type of data**, i.e., int and float. More specifically,
- If we want users to input a number (`int` or `float`), but they input non-digit characters, then your program should keep requesting the data until they provide the appropriate inputs;
 - If we expect an integer, but users input "23#", it is still regarded as a valid user input, as `cin` in C++ can still read "23" correctly from the input stream;
 - You do not need to consider whether the input is semantically meaningful, for example, a negative student ID, a negative math mark, a student name like "\$wang".

The function prototypes for getting valid int input and float input are as follows:

```
float getValidFloat();  
int getValidInt();
```

Below is the main function provided for you.

```
#include <iostream>  
using namespace std;  
  
// Function to get a valid int input  
int getValidInt() {  
    // T0-D0: Write Your Code Here  
}  
  
// Function to get a valid float input  
float getValidFloat() {  
    // T0-D0: Write Your Code Here  
}  
  
int main() {  
    char name[50]; // Student name  
    int studentID; // Student ID  
    float mathMark; // Math mark  
  
    while (true) {  
        // Get student name  
        cout << "Enter student name (or enter '#' to exit): ";  
        cin.getline(name, 50);  
  
        // Check if user wants to exit
```

```

        if (strcmp(name, "#") == 0) {
            break;
        }

        // Get student ID
        cout << "Enter student ID (integer): ";
        studentID = getValidInt();

        // Get math mark
        cout << "Enter math mark (float): ";
        mathMark = getValidFloat();

        // Display student information
        cout << "\nStudent Information:\n";
        cout << "Name: " << name << endl;
        cout << "Student ID: " << studentID << endl;
        cout << "Math Mark: " << mathMark << endl;
        cout << "-----\n";
    }

    cout << "Program exited successfully." << endl;
    return 0;
}

```

Test Cases:

Enter student name (or enter '#' to exit): WANG Yong
 Enter student ID (integer): 23
 Enter math mark (float): 45.67

Student Information:
 Name: WANG Yong
 Student ID: 23
 Math Mark: 45.67

Enter student name (or enter '#' to exit): Andy Heer
 Enter student ID (integer): **www**
 Invalid input! Please enter an integer: **#**
 Invalid input! Please enter an integer: **w234**
 Invalid input! Please enter an integer: **23w**
 Enter math mark (float): **www**
 Invalid input! Please enter a valid float number: **w23.4**
 Invalid input! Please enter a valid float number: **23.4w**

Student Information:
 Name: Andy Heer
 Student ID: **23**
 Math Mark: **23.4**

Enter student name (or enter '#' to exit): **Allen Lee234**

```
Enter student ID (integer): 78.5
Enter math mark (float): 86.5

Student Information:
Name: Allen Lee234
Student ID: 78
Math Mark: 86.5
-----
Enter student name (or enter '#' to exit): #
Program exited successfully.
```

2. **(Versatile Calculation)** Write a C++ function template called `calculate()`, which takes 3 parameters. The first two parameters are numbers of the same type (`int` or `float`). The third parameter is a character (i.e., `char`) representing the operation as given below:
- `+` for addition
 - `-` for subtraction
 - `*` for multiplication
 - `/` for division

The function should return the result of the operation. Also, you should handle the edge case for division by zero by returning 0 if the divisor is 0 (for any numeric type). A function `isZero()` is given for your to check if the divisor is 0.

The main function is given below for your testing. **Note:** For simplicity, we assume users will not input invalid type of data or operator here.

```
#include <iostream>
#include <cmath>
using namespace std;
bool isZero(float num, float epsilon = 1e-6) {
    return fabs(num) < epsilon; // Check if num is very close to 0
}
// T0-D0: Write Your Code Here

int main() {
    cout << "Addition (10 + 5): " << calculate(10, 5, '+') << endl;
    cout << "Subtraction (10.5 - 3.2): " << calculate(10.5, 3.2, '-') << endl;
    cout << "Multiplication (4 * 2): " << calculate(4, 2, '*') << endl;
    cout << "Division (10 / 2): " << calculate(10, 2, '/') << endl;
    cout << "Division (10.6 / 0.0): " << calculate(10.6, 0.0, '/') << endl;
    cout << "Division by zero (10 / 0): " << calculate(10, 0, '/') << endl;
    return 0;
}
```

