

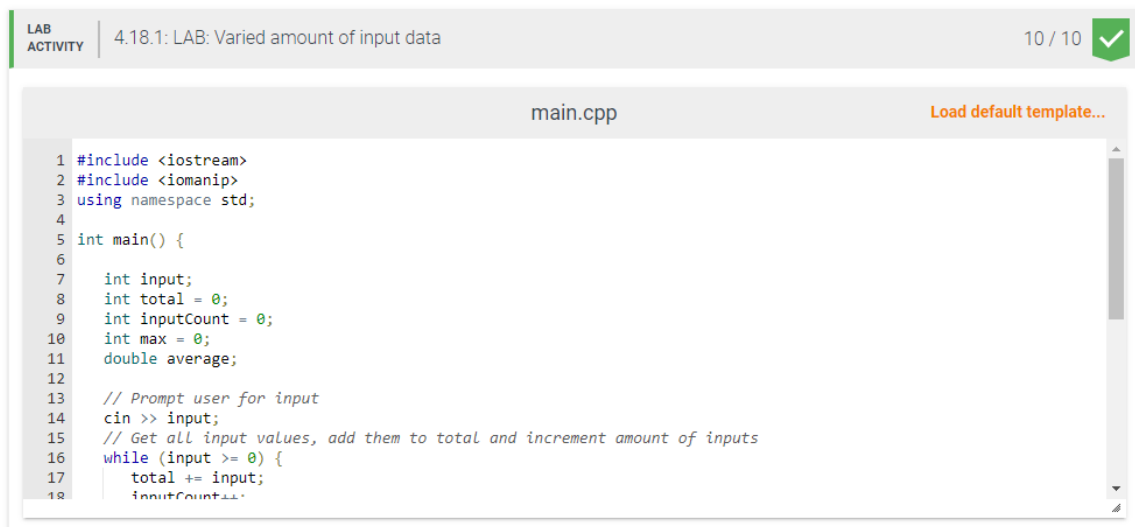
CSCI 140 PA #3 Submission

Due Date: 3/4/23

Name: Ali Mortada

Exercise 1 –zyBook 4.18 LAB: Varied amount of input data

-- check if completely done in zyBook __X__ ; otherwise, discuss issues below
Include a screenshot of current status/score



LAB ACTIVITY | 4.18.1: LAB: Varied amount of input data | 10 / 10 ✓

main.cpp Load default template...

```
1 #include <iostream>
2 #include <iomanip>
3 using namespace std;
4
5 int main() {
6
7     int input;
8     int total = 0;
9     int inputCount = 0;
10    int max = 0;
11    double average;
12
13    // Prompt user for input
14    cin >> input;
15    // Get all input values, add them to total and increment amount of inputs
16    while (input >= 0) {
17        total += input;
18        inputCount++;
```

Exercise 2 –zyBook 4.22 LAB: Password modifier

-- check if completely done in zyBook __X__ ; otherwise, discuss issues below
Include a screenshot of current status/score



LAB ACTIVITY | 4.22.1: LAB: Password modifier | 10 / 10 ✓

main.cpp Load default template...

```
4
5 int main() {
6
7     // Prompt user for input
8     string password;
9     getline(cin, password);
10
11    // Iterate through string, replace old characters with new ones
12    for (int i = 0; i < password.length(); i++) {
13        switch (password[i]) {
14            case 'i':
15            case 'I':
16                password[i] = '1';
17                break;
18            case 'a':
19            case 'A':
20                password[i] = '@';
21                break;
```

Exercise 3 –zyBook 4.29 LAB: Brute force equation solver

-- check if completely done in zyBook __X__ ; otherwise, discuss issues below
Include a screenshot of current status/score

LAB ACTIVITY | 4.29.1: LAB: Brute force equation solver | 10 / 10

main.cpp | Load default template...

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5
6     // Prompt user for input
7     int xCoeffEqn1, yCoeffEqn1, solution1, xCoeffEqn2, yCoeffEqn2, solution2;
8     cin >> xCoeffEqn1 >> yCoeffEqn1 >> solution1 >> xCoeffEqn2 >> yCoeffEqn2 >> solution2;
9
10    // Brute force x and y values from -10 to 10 to see if they solve equation
11    for (int x = 10; x >= -10; x--) {
12        for (int y = 10; y >= -10; y--) {
13            // Check if x and y satisfy both equations and print result if they do
14            if ((x * xCoeffEqn1) + (y * yCoeffEqn1) == solution1 && (x * xCoeffEqn2) + (y * yCoeffEqn2) == solution2)
15                cout << "x = " << x << ", y = " << y << endl;
16            return 0;
17        }
```

Exercise 4 –zyBook 4.30 LAB: Warm up: Drawing a right triangle

-- check if completely done in zyBook __X__ ; otherwise, discuss issues below
Include a screenshot of current status/score

LAB ACTIVITY | 4.30.1: LAB: Warm up: Drawing a right triangle | 3 / 3

main.cpp | Load default template...

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     char triangleChar;
6     int triangleHeight;
7
8     cout << "Enter a character:" << endl;
9     cin >> triangleChar;
10
11    cout << "Enter triangle height:" << endl;
12    cin >> triangleHeight;
13    cout << endl;
14
15    // Print triangle
16    for (int i = 1; i <= triangleHeight; i++) {
17        for (int j = 1; j <= i; j++) {
```

Exercise 5 -- Simple Vending Machine Version 2

-- check if completely done __X__ ; otherwise, discuss issues below

Source code below:

```
/*
    Program: Simple Vending Machine Version 2
    Author: Ali Mortada
    Class: CSCI 140
    Date: 3/15/2023
    Description: User enters amount of quarters, dimes, and nickels to
load vending machine with.
    User then makes purchases using one-dollar bills between 0 and 100
cents, receiving change from coins loaded.
    Program terminates when user enters 0, and outputs number of valid
transactions, remaining amount of coins,
    and total machine balance.
    I certify that the code below is my own work.
    Exception(s): N/A
*/

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

int main() {

    int purchaseAmount, change;
    int numQuarters, numDimes, numNickels, numDollars = 0;
    int validTransactions = 0;
    double machineBalance = 0.00;

    cout << "Vending Machine Version 2 by Ali Mortada" << endl;

    // Prompt user for amount of quarters, dimes, and nickels
    cout << "Enter number of quarters, dimes, and nickels --> ";
    cin >> numQuarters >> numDimes >> numNickels;
    // Repeat number of quarters, dimes, and nickels to user
    cout << "Number of dollars : " << numDollars << endl;
    cout << "Number of quarters: " << numQuarters << endl;
    cout << "Number of dimes : " << numDimes << endl;
    cout << "Number of nickels : " << numNickels << endl;
    // Calculate and output machine balance
    machineBalance += ((numQuarters * 25) + (numDimes * 10) + (numNickels
* 5)) / 100.0;
```

```

    cout << "Machine balance   : $" << fixed << setprecision(2) <<
machineBalance << endl;
    cout << endl << endl;

    cout << "Only one-dollar bills will be accepted." << endl;
    cout << "Only an amount between 0 and 100 is accepted." << endl;
    cout << "Enter 0 to stop the program." << endl;
    cout << endl << endl;

beginningloop:
    // Prompt user for input, store it in purchaseAmount variable
    cout << "Enter a purchase amount [0 - 100] --> ";
    cin >> purchaseAmount;

    // While purchaseAmount != 0 (sentinel value), keep prompting user to
make purchases
    while (purchaseAmount != 0) {
        // Repeat purchase amount to user
        cout << "You entered a purchase amount of " << purchaseAmount << "
cents." << endl;

        // If input amount is invalid, terminate loop and try again
        if (purchaseAmount < 0 || purchaseAmount > 100) {
            cout << "Invalid amount (outside valid range). Try again." <<
endl;

            cout << endl;
            goto beginningloop;
        }

        cout << "Please insert one-dollar bill." << endl;
        cout << "Processing your purchase . . ." << endl;

        // Calculate change and round to nearest multiple of 5
        change = 100 - purchaseAmount;
        change = ((change + 5 / 2) / 5) * 5;

        // If change is more than the amount of coins left, terminate loop
and try again
        if (change > (numQuarters * 25 + numDimes * 10 + numNickels * 5))
        {
            cout << "Insufficient change!" << endl;
            cout << "Your transaction cannot be processed." << endl;
            cout << "Please take back your dollar bill." << endl;
            cout << endl;
            goto beginningloop;
        }
    }

```

```

    }

    // Read amount of change to user
    cout << "Your change of " << change << " cents is given as:" <<
endl;

    int numQuartersNeeded, numDimesNeeded, numNickelsNeeded;
    // Calculate amount of quarters needed
    if (change / 25 >= 1) {
        numQuartersNeeded = change / 25;
        if (numQuartersNeeded > numQuarters)
            numQuartersNeeded = numQuarters;
        change -= (numQuartersNeeded * 25);
    }
    // Calculate amount of dimes needed
    if (change / 10 >= 1) {
        numDimesNeeded = change / 10;
        if (numDimesNeeded > numDimes)
            numDimesNeeded = numDimes;
        change -= (numDimesNeeded * 10);
    }
    // Calculate amount of nickels needed
    if (change / 5 >= 1) {
        numNickelsNeeded = change / 5;
        if (numNickelsNeeded > numNickels)
            numNickelsNeeded = numNickels;
        change -= (numNickelsNeeded * 5);
    }

    // Output change
    cout << "    quarter(s): " << numQuartersNeeded << endl;
    cout << "    dime(s)   : " << numDimesNeeded << endl;
    cout << "    nickel(s) : " << numNickelsNeeded << endl;

    // Remove change from machine
    numQuarters -= numQuartersNeeded;
    numDimes -= numDimesNeeded;
    numNickels -= numNickelsNeeded;

    // Update number of valid transactions and machine balance
    validTransactions++;
    machineBalance = ((numQuarters * 25) + (numDimes * 10) +
(numNickels * 5)) / 100.0;

```

```

        // Prompt user again for input before redoing the loop (prevents
infinite loop)
        cout << endl;
        cout << "Enter a purchase amount [0 - 100] --> ";
        cin >> purchaseAmount;
    }

    // Print valid transactions, number of each type of coin and machine
balance
    cout << "There are " << validTransactions << " valid transactions." <<
endl;
    cout << "Number of dollars : " << validTransactions << endl;
    cout << "Number of quarters: " << numQuarters << endl;
    cout << "Number of dimes   : " << numDimes << endl;
    cout << "Number of nickels : " << numNickels << endl;
    cout << "Machine balance   : $" << fixed << setprecision(2) <<
machineBalance + validTransactions << endl;

    return 0;
}

```

Input/output below:

```
Vending Machine Version 2 by Ali Mortada
Enter number of quarters, dimes, and nickels --> 2 4 4
Number of dollars : 0
Number of quarters: 2
Number of dimes   : 4
Number of nickels : 4
Machine balance   : $1.10
```

Only one-dollar bills will be accepted.
Only an amount between 0 and 100 is accepted.
Enter 0 to stop the program.

```
Enter a purchase amount [0 - 100] --> 36
You entered a purchase amount of 36 cents.
Please insert one-dollar bill.
Processing your purchase . . .
Your change of 65 cents is given as:
    quarter(s): 2
    dime(s)   : 1
    nickel(s) : 1
```

```
Enter a purchase amount [0 - 100] --> -1
You entered a purchase amount of -1 cents.
Invalid amount (outside valid range). Try again.
```

```
Enter a purchase amount [0 - 100] --> 35
You entered a purchase amount of 35 cents.
Please insert one-dollar bill.
Processing your purchase . . .
Insufficient change!
Your transaction cannot be processed.
Please take back your dollar bill.
```

```
Enter a purchase amount [0 - 100] --> 75
You entered a purchase amount of 75 cents.
Please insert one-dollar bill.
Processing your purchase . . .
Your change of 25 cents is given as:
    quarter(s): 0
    dime(s)   : 2
    nickel(s) : 1
```

```
Enter a purchase amount [0 - 100] --> 105
You entered a purchase amount of 105 cents.
Invalid amount (outside valid range). Try again.
```

```
Enter a purchase amount [0 - 100] --> 0
There are 2 valid transactions.
Number of dollars : 2
Number of quarters: 0
Number of dimes   : 1
Number of nickels : 2
Machine balance   : $2.20
```

Answer for Question 1

It is best to use a while loop over a for loop when you want the loop to end based on a condition, since a while loop checks a condition first and then executes the code. A for loop would be better if you just want the loop to iterate a fixed number of times.

Answer for Question 2

D: running the loop forever can be ignored, since the program would not terminate if it had an infinite loop. This is usually something that the programmer does not desire, and could even cause the program to crash if it is not manually terminated by the user.

Extra Credit – provide if applicable.