EC327 – "Introduction to Software Engineering"

Lab 4 – Functions

The goal of Lab 4 is to solve larger problems with little to no code in main.cpp and be able to use functions and organize them in different, external files and compile them together to make one executable.

To complete Lab4 please **SOLVE ONE OF THE FOLLOWING PROBLEMS.**

For the problems in this lab, we are asking you to implement and utilize:

- Data types
- User input
- Conditional branches – if, else, if else
- Loops – do-while, for, while
- Nested loops
- Functions
- Header files

During your designated lab hours, you should be able to demonstrate and explain what your solution to your chosen problem is. You are not required to work outside of the lab hours. We recommend starting to work early and at a minimum reading and understanding the lab before you come in. You do not need to code the program before lab, but it is highly encouraged for you to have an understanding of what you are going to do to complete the lab. For example, you can solve the problem at home, come to your lab session, demonstrate your solution, and then work on other related assignments (HWs, PAs, etc.). But, ensure that your code compiles and executes properly in the lab environment!

Read carefully the problem definitions. **Make your own decision** on what problem to solve. **Decide for yourself** when to start because on the exams you are on your own. But in the labs we are **happy to answer questions.**

Good Luck!

P1 – Quadratic solver

Lab4P1.cpp xFactor.cpp xFactor.h

In this assignment, you are expected to create three files. Lab4P1.cpp will have the main function. xFactor.cpp will have the functions. xFactor.h will have the function prototypes

Do not include the <cmath>, that's cheating.
An easy way to calculate the square root of N is
guess = (guess + N / guess) / 2
and then make sure that is within your tolerance
tol >= abs(guess – N / guess) / min(guess, N/ guess)

Use this and the quadratic and solve for the positive and negative x values and print them out for the user.

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Function Prototypes:

double sqrt(double N, double tol);
double positiveX(int a, int b, int c, int tol);
double negative(int a, int b, int c, int tol);

bash$ ./Lab4P1
What are the coefficients of your quadratic? 1 8 12
What is the tolerance you would like? .0001
x = (-2, -6)
Again? (y/n) y
What are the coefficients of your quadratic? 1 5 6
What is the tolerance you would like? .0001
x = (-2, -3)
Again? (y/n) y
What are the coefficients of your quadratic? 4 10 6
What is the tolerance you would like? 0.0001
x = (-1, -1.5)
Again? (y/n) n
Program ends

RED IS USER INPUT We will not check for imaginary input, but you could try to solve that for extra practice

P2 – Counting Heads

Lab4P2.cpp countHeads.cpp countHeads.h

Given N coin flips, the number of ways one can get r heads is given by $\binom{n}{r} = \frac{n!}{r!(n-r)!}$

n! is the factorial function defined as 1 x 2 x 3 x … x (n-1) x n. Write a function that takes in n and r as non-negative integers and n >= k. Use this expression to count the total number of ways one can get at least r out of n heads.

Be careful. Factorials scale quick and will be out of range of data types fast.

Function prototypes:

int choose(int N, int r);

```
bash$ ./Lab4P2
How many coins would you like to flip? 5
How many heads do you want? 2
Flipping 5 coins, we can get 2 heads 10 ways.
Flip again? (y/n) y
How many coins would you like to flip? 4
How many heads do you want? 2
Flipping 4 coins, we can get 2 heads 6 ways.
Flip again? (y/n) y
How many coins would you like to flip? 1000
How many heads do you want? 999
Flipping 1000 coins, we can get 999 heads 1000 ways.
Flip again? (y/n) n
Program ends
```

RED IS USER INPUT. Make sure this works for big numbers.

P3 – Cashier Denominations

Lab4P3.cpp denominations.cpp denominations.h

In this problem, we will help speed up transactions for cashiers in a grocery store. You will make a program for the cashier to quickly determine the changes they should give back to a customer's purchase. The user will enter the total checkout amount (value of the goods he/ she bought). The user will then enter the amount they tender (payment made). The program will return the number of dollar bills, quarters, dimes, nickels, and pennies that the cashier should return to the customer. The function dollarbills should take the total cost amount, and total amount tendered, and return the number of dollar bills that should be returned as change. Similarly, the function quarters should return the number of quarters that should be returned as part of the change. The function nickels, and quarters work the same way. Finally, your program should output the cashier the total number of dollar bills, quarters, dimes, nickels, and pennies they should return to the customer to complete the transaction.

Remember the store only uses dollar bills, and the coins discussed above and there are no bills of 5$, 10$, 100$.

Function prototypes:

double dollarbills(double cost, double tendered);

double quarters(double cost, double tendered);

double dimes(double cost, double tendered);

double nickels(double cost, double tendered);

double pennies(double cost, double tendered);

Example 1 :

bash$ ./Lab4P3
What is the checkout amount ? $32.87
What is the amount tendered ? $50
Output : **You should return 17 dollar bills, 3 quarters, 1 dime, and 2 nickels.**

Example 2 :

bash$ ./Lab4P3

What is the checkout amount ? $5.44
What is the amount tendered ? $4
Output : **You should return 1 dollar bills, 1 quarters, 1 dime, 1 penny, and 4 nickels.**

RED IS USER INPUT.