Midterm 1 Performance

Midterm 1 is out of 40.

26+ in A range (A- to A+)

18+ in B range (B- to B+)

10+ in C range (C- to C+)

lower than 10 D
Functions

Recall basic arithmetic operations on int.
+ , - , * , / , %

How do we compute square root? There’s no basic operation, so the programmer has to figure out and write the code.

But it’s such a useful operation, does that mean, every program that needs to compute some square root, has to provide the code?

Solution – Reusable Code
- Once someone figures out how to accomplish a specific task and write it out, you don’t have to figure it out again.
- Write the function just once, then execute it whenever you need to perform the task.
Functions – Cont’

To make our life easier, task like to compute square root, has already been written and saved in a library – math. Each saved code that performs a specific task called function.

To execute the saved code – to call a function.

Example: function sqrt to compute square root

```cpp
double n = 4.0;
double ans_sqrt_of_n = sqrt(n);
cout << ans_sqrt_of_n << endl;
```

The output is 2 (note cout by default doesn’t print out tailing 0s after decimal point)
Functions

Takes – 0 or more inputs (arguments)
Returns – 0 or 1 outputs (return value)

When call a function, call by it’s name, pass in arguments, and expect to use the return value.

double ans_sqrt_of_n = sqrt(n);
Functions in C++ Files

Some useful functions has already been provide (pre-defined) in C++ files.

sqrt in cmath
- takes a double argument
- return a double, square root of the argument

rand in cstdlib
- doesn’t take any arguments
- returns a random integer.

srand in cstdlib
time in ctime

In order to call these pre-defined functions, we have to include it’s file in the program
/* ask user for three values a and b 
 * print out the roots for quadratic equation */
#include <cmath>
#include <iostream>
using namespace std;
int main(){
    int a, b, c;
    cout<<"Enter a, b and c values for quadratic: ";
    cin >> a >> b >> c;
    /* recall formula to find roots 
        * let’s call radicand d 
        * no roots if a is 0 or d is negative*/
    int d = b * b - 4 * a * c;
    if (a == 0 || d < 0) {
        cout << "No roots."
    } else {
        int r1 = (-1 * b + sqrt(d)) / (2 * a);
        int r2 = (-b - sqrt(d)) / (2 * a);
        cout << "Roots: ", r1 << ",", r2 << endl;
        return 0;
    }
return 0; }