TIC1001—Introduction to Computing and Programming
National University of Singapore

Practice on Basic C Expressions

BMI

The body mass index (BMI), or Quetelet index, is a statistical measure of body weight based on a person's weight and height. Though it does not actually measure the percentage of body fat, it is used to estimate a healthy body weight based on a person's height.

A person's BMI is given by the following formula: $BMI = \frac{mass(kh)}{(height(m))^2}$

Complete the function bmi that takes in mass (g) and height (cm) and returns the BMI.

In [ ]:
```
float bmi(int mass, int height)
{
    float bmi;
    bmi = (mass / 1000.0) / ((height / 100.0)*(height / 100.0));
    return bmi;
}
```

Hypothenuse

The hypothenuse of a right-angled triangle can be computed using Pythagoras theorem.
Complete the function hypothenuse which takes in the length of 2 sides of a right-angle triangle and returns the length of the hypothenuse.
You may use the sqrt function available in math.h

In [ ]:
```
#include <math.h>
double hypothenuse(double a, double b)
{
    double hypo, sides;
    sides = (a * a) + (b * b);
    hypo = sqrt(sides);
    return hypo;
}
```

Investment

If you invest a principal amount of P dollars at R percent interest rate compounded annually, in N years, your investment will grow to $\frac{P[1-(\frac{R}{100})^{N+1}]}{1-\frac{R}{100}}$ dollars.

Implement a function `investment` that accepts positive integers p, r and n and computes the amount of money earned after n years. You may assume that the interest rate is always smaller than 100.

In [ ]:
```
#include <math.h>
double investment(int p, int r, int n)
{
    double total;
    total = p * (1 - pow((r / 100.0),n + 1)) / (1 - (r / 100.0));
    return total;
}
```

Root of Linear Equation

In algebra, we have studied that second degree linear equations always have two roots.
Implement a function `get_bigger_root` that reads three integer coefficients a, b, c representing equation ax2 + bx + c = 0. This function returns the bigger one between its two roots (assuming that both roots are real numbers and that a is always positive).

In [ ]:
```
#import <math.h>

double get_bigger_root(int a, int b, int c)
{
    double root1, root2, root_large, b_sq_minus_4ac;
    //Formula to find root
    b_sq_minus_4ac = (b * b) - (4 * a * c);
    root1 = (- b + sqrt(b_sq_minus_4ac)) / (2 * a);
    root2 = (-b - sqrt(b_sq_minus_4ac)) / (2 * a);
    //Determine larger root
    if (root1 > root2)
    {
        root_large = root1;
    }
    else
    {
        root_large = root2;
    }
    return root_large;
}
```

nth Digit

Implement a function `nth_digit` that takes as inputs two integers, num and n. The function should return the nth digit of num from the right.

In [ ]:
```
#include <math.h>
int nth_digit(int num, int n)
{
    //nth digit = (num % 10^n) / 10^n-1
    int digitn = pow(10,n);
    int digitn_del= pow(10, n-1);
    int results = (num % digitn) / (digitn_del);

    //If out of range
    if (digitn > num)
    {
        results = 0;
    }
    else
    {
        results = results;
    }

    return results;
}
```

-END-