Function Return Values

1. What is the output of the following program

```
#include <stdio.h>

int fn1(int x);
void fn2();
int fn3(int x);
void fn4();
void fn5();

int main( void ) {
    fn2();
    printf("---\n");
    fn4();
    printf("---\n");
    fn5();
    printf("---\n");

    return 0;
}

int fn1(int x) {
    printf( "fn1: %d\n", x);
    x++;
    printf( "fn1: %d\n", x);
    return x;
}

void fn2( ) {
    int y = 5;
    printf( "fn2: %d\n", y);
    y = fn1(y);
    printf ( "fn2: %d\n", y);
}

int fn3(int x) {
    printf( "fn3: %d\n", x);
    x *= 2;
    printf( "fn3: %d\n", x);
    return x;
}

void fn4( ) {
    int x = 6;
    printf( "fn4: %d\n", x);
    fn3(x);
    printf( "fn4: %d\n", x);
}

void fn5( ) {
    int x = 6;
    printf( "fn5: %d\n", x);
    x = fn3(x);
    printf( "fn5: %d\n", x);
}
```

OUTPUT:

```
fn2: 5
fn1: 5
fn1: 6
fn2: 6
---
fn4: 6
fn3: 6
fn3: 12
fn4: 6
---
fn5: 6
fn3: 6
fn3: 12
fn5: 12
---
```

2. Consider the function `mystery` below. Complete the documentation and give it a better name. Within the main method provided, write code to test this function – that is, call the function and print out the result of the call.

```
#include <stdio.h>

int get_double_x(int x);
int main( void ) {
    int num = 5;
    int result = get_double_x(num);
    printf("double of %d is: %d\n", num, result); //double of 5 is: 10

    return 0;
}

/*
 * Purpose: computes the value of 2 * x
 * Parameters: int x
 * Returns: int, double the value of x
 */
int get_double_x(int x) {
    x *= 2;
    return x;
}
```

3. Design a function that takes the amount of time an object takes to fall after being dropped in seconds. The function calculates and returns the distance the object fell where the formula for distance is:
   $d = \frac{1}{2} gt^2$ where t is the time and g is gravitational acceleration is constant at 9.8 m/s^2.
   Your function should assume time is not a negative number.
   Write the code to test your function. Don't forget the function prototype and documentation!
4. Design a function that takes the distance in meters from a point to the ground and a time in seconds. The function should determine whether the object will hit the ground in the given time. It should return 1 if it will hit the ground and 0 otherwise.
   The function should assume that both time and distance are not negative.
   Think carefully about how this function can and should use the function from Question 3.

```
#include <stdio.h>
#define ACCEL 9.8

double fall_distance(int time);
int does_reach_ground(double height, int time);

int main( void ) {
    double dist;
    int reached_ground;

    dist = fall_distance(0);
    printf("%.1f\n", dist); // should print 0.0
    dist = fall_distance(104);
    printf("%.1f\n", dist); // should print 52998.4

    reached_ground = does_reach_ground(4410, 30);
    printf("%d\n", reached_ground); // should print 1
    reached_ground = does_reach_ground(4409, 30);
    printf("%d\n", reached_ground); // should print 1
    reached_ground = does_reach_ground(3063, 25);
    printf("%d\n", reached_ground); // should print 0

    return 0;
}

/*
 * Purpose: calculates the distance an object falls in given time
 * Parameters: int time – drop time in seconds, >=0
 * Returns: double - distance
 */
double fall_distance (int time) {
    double dist = ACCEL * time * time / 2;
    return dist;
}

/*
 * Purpose: determines if an object falling a given time will hit the ground
 * note: result is approximate given inaccuracies of floating point arithmetic
 * Parameters: double height - height the object is dropped from in meters, >=0
 *      int time – drop time in seconds, >=0
 * Returns: int, 1 if it reaches the ground, 0 otherwise
 */
int does_reach_ground(double height, int time){
    double distance_fallen;
    distance_fallen = fall_distance(time);
    return distance_fallen >= height;
}
```

5. Design a function that takes the following arguments: the specific gravity of crude oil (in kg/cubic metre) and the number of barrels of crude oil purchased at this specific gravity. Assume these values are not integers. Your function should convert the specific gravity to the American Petroleum Institute's gravity scale in "degrees API". The formula for converting between specific gravity and API Gravity is:

   API Gravity = 141.5 / (specific gravity / 1000) – 131.5

   Your program must also compute the amount of crude oil purchased in cubic metres. Note that one barrel of crude oil = 0.158910 cubic metres.

   Your function must then print a message depending on the computed value of API gravity:

   If it's more than 31.1 degrees API, print: "**You purchased xx.xx cubic metres of Light Crude**".

   If it's more than 22.3 degrees API, but not more than 31.1 degrees, print: "**You purchased xx.xx cubic metres of Medium Crude**".

   Otherwise, print out "**You purchased xx.xx cubic metres of Heavy Crude**".

   where xx.xx is the number of cubic metres of crude oil purchased, to 2 decimal places.

   After printing the message, the function should return the value of API Gravity that was computed.

```
#include <stdio.h>

#define CUBIC_METERS_PER_BARREL 0.158910
#define LIGHT_CRUDE_MIN         31.1
#define MEDIUM_CRUDE_MIN        22.3

double get_api_gravity(double specific_gravity, double num_barrels);

int main(void) {
    double api;

    printf("should print: 0.32 cubic meters of Light Crude and return API ~45.38\n");
    api = get_api_gravity(800, 2);
    printf("api: %f degrees API\n", api);

    printf("\nshould print: ~0.16 cubic meters of Medium Crude and return API ~25.72\n");
    api = get_api_gravity(900, 1);
    printf("api: %f degrees API\n", api);

    printf("\nshould print: ~0.79 cubic meters of Heavy Crude and return API ~10.00\n");
    api = get_api_gravity(1000, 5);
    printf("api: %f degrees API\n", api);

    return 0;
}
/*
 * Purpose: calculates api gravity using specific gravity
 *   calculates amount of crude given num_barrels
 *   prints a message with amount and type of crude
 * Parameters: double specific_gravity, in kg/cubic meter
 *             double num_barrels, number of barrels of crude oil
 * Returns: double - api gravity
 */
double get_api_gravity(double specific_gravity, double num_barrels){
    double api_gravity = 141.5 / (specific_gravity / 1000) - 131.5;

    double amount_crude = CUBIC_METERS_PER_BARREL * num_barrels;
    printf("Your purchased %.2f cubic meters of ", amount_crude);

    if(api_gravity > LIGHT_CRUDE_MIN) {
        printf("Light Crude\n");
    } else if(api_gravity > MEDIUM_CRUDE_MIN) {
        printf("Medium Crude\n");
    } else {
        printf("Heavy Crude\n");
    }

    return api_gravity;
}
```

6. Write a function to compute and return the income tax based on a person's income passed as an argument to the function. Be sure to make use of constants use as few conditions as possible!

Tax is calculated as follows:

if the income is ≤ 12500 tax payable is 0%

if the income is higher than 12500 but less than 65000, tax payable is 24% of only the income over $12,500

if the income is higher than 65000 the tax payable is $12,600 plus 32% of only income over $65,000

```
#include <stdio.h>

#define BASE_CUTOFF   12500.0
#define MID_CUTOFF    65000.0
#define MID_RATE      0.24
#define HIGH_RATE     0.32
#define HIGH_BASE_TAX 12600.0

double calc_tax(double income);

int main( void ) {
    // calls to test the function
    double tax;

    printf("\nshould be ~\n0.00\n");
    tax = calc_tax(BASE_CUTOFF - 1);
    printf("%.2f\n", tax);

    printf("\nshould be ~\n0.00\n");
    tax = calc_tax(BASE_CUTOFF);
    printf("%.2f\n", tax);

    printf("\nshould be ~\n0.24\n");
    tax = calc_tax(BASE_CUTOFF + 1);
    printf("%.2f\n", tax);

    printf("\nshould be ~\n12599.76\n");
    tax = calc_tax(MID_CUTOFF - 1);
    printf("%.2f\n", tax);

    printf("\nshould be ~\n12600.00\n");
    tax = calc_tax(MID_CUTOFF);
    printf("%.2f\n", tax);

    printf("\nshould be ~\n12600.32\n");
    tax = calc_tax(MID_CUTOFF + 1);
    printf("%.2f\n", tax);

    return 0;
}

/*
 * Purpose: calculates the tax owing based on given income
 * Parameters: double income – a person's year income, >=0
 * Returns: double – tax owing
 */
double calc_tax(double income) {
    double tax;

    if ( income <= BASE_CUTOFF )
        tax = 0;
    else if ( income <= MID_CUTOFF )
        tax = ( income - BASE_CUTOFF ) * MID_RATE;

    else
        tax = HIGH_BASE_TAX + ( income - MID_CUTOFF ) * HIGH_RATE;

    return tax;
}
```