a)

```
int sum(int x, int y) {
    int result;
    result = x + y;
    // result is a local variable. Calling function cannot use the result because it was never
returned. Add return statement.
    return result;
}
```

b)

```
int sum(int n) {
    if (0 == n) { // it is best practice to always use brackets even for one statement.
        return 0;
    } else {
        n = n + n;
    }
    // n is local parameter. The calling function's argument never changed. Add return statement
    return n;
}
```

c)

```
double x = 1E10;
cout << "square of 1E10 = " << square(x) << endl;
// in main program, x is data type double. It will be truncated when the square function is
invoked and incorrect values will be produced. Promotion rules were not followed. Cannot safely
convert double to integer. Fix by changing argument and return value for Square function to data
type double .
double square(double x) {
    return x * x;
}
```

d)

```
int main()  {
    int i = 0;
    for(i = 0; i < 3; i++) // logical error : remove a semicolon after loop .
    {
        std::cout << i << std::endl;
        continue;
    }
    return 0;
}
```

e)

```
#include<iostream>

void main() // linker error: Here Main() should be main()
{
    int a = 10;
    std::cout << a; // compile error: missing semicolon
}
```

f)

```
// based on
// https://stackoverflow.com/questions/31495207/c-function-exercise-keeps-returning-zero

#include <iostream>
using namespace std;
// bad and obsucure name - better one is hailstone_sequence_length (not count_something)
int hailstone_sequence_length(int x) {
    int counter = 0; // not initialized local variables have undefined value (could be any)

    while (x != 1) { // loop should continue until x is not 1. Also using while is more
appropriate .
        if (x % 2 == 0) {
            x = x / 2;
            counter++;
        } else if (x % 2 != 0) {
            x = x * 3 + 1;
            counter++;
        }
    }
    return counter;
}

int main() {
    int x = 10, counter;
    // You need to put the return value of hailstone into a 'counter' variable
    counter = hailstone_sequence_length(x);
    cout << counter << endl;
    return 0;
}
```

g)

```
// return type must be bool. Bool type must be used when expressing binary condition
bool isPrime(int n) {
    // we do not need to declare i twice
    // Also it's best practice to declare variable to have as little scope as possible
    // hence defining it in for initialization part (i not visible outside of loop)
    for (int i = 2; i < n; i++)
        // i cannot be zero (0) - dividing by zero is runtime error (program crashes)
        // i cannot be one (1) - dividng (modulus) by 1 is always 0 and program will errenously
return that no numbers are prime
        // hence loop start from 2
        if (n % i == 0) {
            return 0;
        }
    return 1;
}
void check(int numbers[], int finish) {
    for (i = 0; i < finish; i++)
        // we must check if the i-th number in the array is odd  and not the index
        if (!isPrime(numbers[i])) {
            std::cout << numbers[i] << "is prime\r\n";
        }
}
```

h)

```
#include <iostream>
// missing namespace std
// though generally it's bad practice to declare namespace with using namespace
// see why here:
// https://stackoverflow.com/questions/1452721/why-is-using-namespace-std-considered-bad-practice
using namespace std;

int main(int argc, const char *argv[]) {
   char passWordInput[100];
   // this is declaration of the function and not invocation
   // hence the function is not executed.
   void mainMenu();
   mainMenu();
   // cin.getline is allowed to read 200 while input size is 100
   // This is very dangerous since it may overwrite other variables or program will crash
   cin.getline(passWordInput, sizeof(passWordInput));
   cout << strlen(passWordInput);
   return 0;
}
void mainMenu() {
   cout << "Your password must be at least 8 characters long and" << endl;
   cout << "contain at least one character from the following categories." << endl;
}
```