

## In class Practice 9/20/2019

Group three or four people team.

For Part I questions, do solo first then discuss as a group.

For Part II and part III work as a team

### Part I

1. A variable known only within the function in which it's defined is called a(n) local variable .
2. All the functions in <cmath> are global functions
3. To get remainder of two floating point numbers, one can use fmod function in <cmath>
4. The keyword void is used in a function header to indicate that a function does not return a value or to indicate that a function contains no parameters.
5. An identifier's scope is the portion of the program in which the identifier can be used.
6. A(n) function prototype allows the compiler to check the number, types and order of the arguments passed to a function.
7. Function rand is used to produce random numbers.
8. Function srand is used to set the random-number seed to randomize the number sequence generated by function rand.
9. For a local variable in a function to retain its value between calls to the function, it must be declared static
10. A function that calls itself either directly or indirectly (i.e., through another function) is a(n) recursive function.
11. It's possible to have various functions with the same name that operate on different types or numbers of arguments. This is called function overloading.
12. The const qualifier is used to declare read-only variables.

## Part II

Find the errors of the following program segments

1.

```
int sum(int n) { // assume n is nonnegative
    if (0 == n) return 0;
    else n + sum(n - 1);      else return n+sum(n-1);
}
```

2.

```
void product() {
    int a{0};
    int b{0};
    int c{0};
    cout << "Enter three integers: ";
    cin >> a >> b >> c;
    int result{a * b * c};
    cout << "Result is " << result;
    return result;      delete "return result"
}
```

3.

```
void f1() {
    cout << "Inside function f1" << endl;
    void f2() {
        cout << "Inside function f2" << endl;
    }
}
```

move the definition of f2 out of the scope of definition of f1

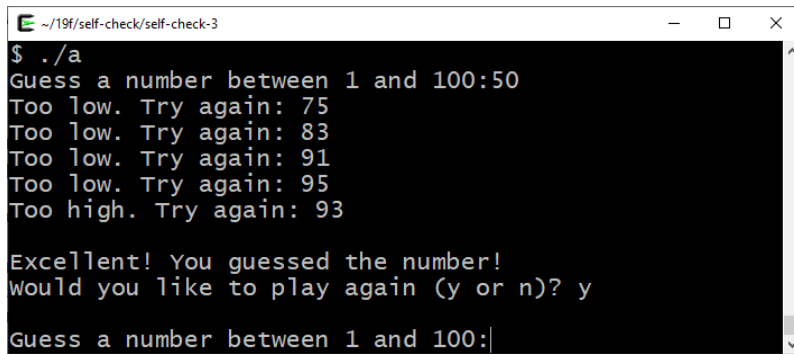
### Part III

#### Q1

Use `rand()` and `srand` to create a random number and ask the user to guess this number. Assume the number fall between 1 and 100. If user guessed the number right, your program will ask if the user want to continue to play the guessing game. If answer is 'y', game continue, otherwise quit.

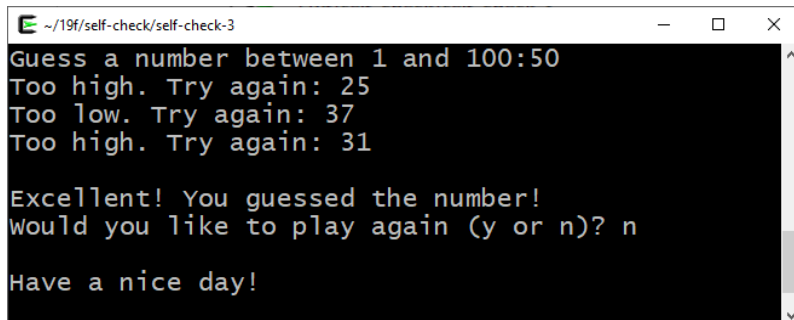
During the process of guessing game, if user guessed wrong, your program will give a hint such as "Too high" or "Too low".

The following screen shots show a test runs.



```
~/19f/self-check/self-check-3
$ ./a
Guess a number between 1 and 100:50
Too low. Try again: 75
Too low. Try again: 83
Too low. Try again: 91
Too low. Try again: 95
Too high. Try again: 93

Excellent! You guessed the number!
would you like to play again (y or n)? y
Guess a number between 1 and 100:|
```



```
~/19f/self-check/self-check-3
Guess a number between 1 and 100:50
Too high. Try again: 25
Too low. Try again: 37
Too high. Try again: 31

Excellent! You guessed the number!
would you like to play again (y or n)? n

Have a nice day!
```

Q2

Write a recursive function to solve Ackermann function. A wiki page of Ackermann function can be found in [https://en.wikipedia.org/wiki/Ackermann\\_function](https://en.wikipedia.org/wiki/Ackermann_function)

$$A(m, n) = \begin{cases} n + 1 & \text{if } m = 0 \\ A(m - 1, 1) & \text{if } m > 0 \text{ and } n = 0 \\ A(m - 1, A(m, n - 1)) & \text{if } m > 0 \text{ and } n > 0. \end{cases}$$

The function result is fairly big even when m and n are small. A test run caused segmentation fault when m=3 and n = 13.

```
~/19f/self-check/self-check-3
Ackermann (2, 0) is 3
Ackermann (2, 1) is 5
Ackermann (2, 2) is 7
Ackermann (2, 3) is 9
Ackermann (2, 4) is 11
Ackermann (2, 5) is 13
Ackermann (2, 6) is 15
Ackermann (2, 7) is 17
Ackermann (2, 8) is 19
Ackermann (2, 9) is 21
Ackermann (2, 10) is 23
Ackermann (2, 11) is 25
Ackermann (2, 12) is 27
Ackermann (2, 13) is 29
Ackermann (2, 14) is 31
Ackermann (3, 0) is 5
Ackermann (3, 1) is 13
Ackermann (3, 2) is 29
Ackermann (3, 3) is 61
Ackermann (3, 4) is 125
Ackermann (3, 5) is 253
Ackermann (3, 6) is 509
Ackermann (3, 7) is 1021
Ackermann (3, 8) is 2045
Ackermann (3, 9) is 4093
Ackermann (3, 10) is 8189
Ackermann (3, 11) is 16381
Ackermann (3, 12) is 32765
segmentation fault (core dumped)
```