C Programming lab Functions quiz

Name: ____________________

Email: ____________________

Roll nos: ____________________

Concept: *parts of a function*

1. What is the declaration of this function?

```
int almostSquare(int x,int y)
    {
    return (x - 1) * (y + 1);
    }
```

   (a) `int almostSquare(int x,int y);`
   (b) `int almostSquare(int , int);`
   (c) `return (x - 1) * (y + 1);`
   (d) `int almostSquare(x,y){ return (x-1) * (y+1)}`

2. What is the declaration of this function?

```
int f()
    {
    return 1;
    }
```

   (a) `int f() {return 1;}`
   (b) `int f(void);`
   (c) it has no declaration
   (d) `int f();`

3. What is the body of this function?

```
int almostSquare(int x,int y)
    {
    return (x - 1) * (y + 1);
    }
```

   (a) `int almostSquare(x,y)`
   (b) `return times`
   (c) `int almostSquare()`
   (d) `return (x - 1) * (y + 1);`

4. What is the body of this function?

```
int f(x)
    {
    return g(x * 2);
    }
```

(a) `return g(x * 2);`

(b) `return function call`

(c) `return g;`

(d) `return f(g(x * 2));`

5. What is this?

```
square(x);
```

(a) a function definition

(b) a function program

(c) a function body

(d) a function call

6. What is this?

```
int square(int x)
    {
    return x * x;
    }
```

(a) a function body

(b) a function definition

(c) a function program

(d) a function call

7. What is this?

```
int almostSquare(x)
    {
    return (x + 1) * (x - 1);
    }
```

(a) a function call

(b) a complete program

(c) a function definition

8. What is this?

```
square(x);
```

(a) A function definition

(b) A function body

(c) A function call

(d) A function program

9. What is this?

```
almostSquare(a);
```

(a) a function definition
(b) a complete program
(c) a function call

Concept: *recognizing complete programs*

10. What is this?

```
#include<stdio.h>
int almostSquare(x)
    {
    return (x + 1) * (x - 1);
    }
int main()
    {
    printf("%d",almostSquare(5));
    }
```

(a) a function definition
(b) a function call
(c) a function declaration
(d) a complete program

11. Is this a complete program?

```
#include<stdio.h>
int main()
    {
    int x = 5
    printf("x is %d\n",x);
    return 0;
    }
```

(a) no, *main* is called, but not defined
(b) yes, the program prints correctly
(c) no, *main* is defined, but not called

12. Is this a complete program?

```
#include<stdio.h>
int main()
    {
    int x = 5;
    printf("x is %d\n",x);
    return 0;
    }
int funtest()
    {
    main();
    }
```

(a) no, *main* is called from funtest
(b) no, *main* is defined, but not called
(c) yes, the program prints correctly

(d) no, this program execution results in segmentation fault

(e) no, *main* is called, but not defined

13. Is this function definition correct?

```
int intent(int x,int y)
    {
    if(y == 0)
        return 0;
    return 1 + intent(x,y-1)
    }
```

(a) yes, it is correct as written

(b) no, a semicolon character is missing

(c) no, there should only one formal parameter

14. Is this function definition correct?

```
int intent(int x,int y)
    {
    if(y == 0)
        return 0;
    return 1 + intent(x,y-1);
    }
```

(a) yes, it is correct as written

(b) no, a semicolon character is missing

(c) no, there should only one formal parameter

15. Is this function definition correct?

```
void intent(int x,int y)
    {
    if(y == 0)
        return 0;
    return 1 + intent(x,y-1);
    }
```

(a) no, there should only one formal parameter

(b) no, cannot call function within a function

(c) yes, it is correct as written

(d) no, return value is void and still trying to return int datatype

16. Is this function definition correct?

```
void if(int x,int y)
    {
    printf("%d\n",x * y);
    return;
    }
```

(a) no, there is no return value

(b) yes, it is correct as written

(c) no, you cannot print inside a function

(d) no, the name of the function is a keyword

(e) error: invalid use of void expression

17. Is this function definition correct?

```
int debug = 1;
void debug(char *message,int value)
    {
    if (debug)
        printf(message,value);
    return;
    }
```

(a) error: invalid use of void expression
(b) no, debug variable redeclared
(c) no, the name of the function is a keyword
(d) no, there is no return value
(e) yes, it is correct as written
(f) no, there is no `else` keyword

18. Is this function definition correct?

```
int debug = 1;
void debugMsg(char *message,int value)
    {
    if (debug)
        printf(message,value);
    return;
    }
```

(a) yes, it is correct as written
(b) no, there is no return value
(c) no, the name of the function is a keyword
(d) no, there is no `else` keyword
(e) error: invalid use of void expression
(f) no, debug redeclared

Concept: *matching calls and definitions*

19. Does the function call match the function definition?

```
int square(int x)
    {
    return x * x;
    }

int main()
    {
    int result;
    result = square();
    return 1;
    }
```

(a) no, there are too few formal parameters
(b) yes

(c) no, there are too few arguments

(d) no, there are too many arguments

20. Does the function call match the function definition?

```
int square(int x)
    {
    return x * x;
    }

int main()
    {
    int result;
    result = square(3);
    return 1;
    }
```

(a) no, there are too many arguments

(b) no, the function call should pass the variable x

(c) yes

(d) no, there are too few formal parameters

21. Does the function call match the function definition?

```
int square(int x)
    {
    return x * x;
    }

int main()
    {
    int a = 3;
    result = square(a);
    return 1;
    }
```

(a) no, there are too many arguments in the call

(b) no, there are too few formal parameters in the definition

(c) no, the function call should pass the variable x, not a

(d) yes

(e) no, the function call should pass a value, not a variable

22. Does the function call match the function definition?

```
int square(int x)
    {
    return x * x;
    }
int main()
    {
    int result;
    result = square(3,7);
    }
```

(a) no, there are too few formal parameters in the definition

(b) no, the function call should pass the variable x twice

(c) no, there are too many arguments in the call

(d) yes

23. Does the function call match the function definition?

```
int almostProduct(a,b)
    {
    return a * (b - 1);
    }
int main()
    {
    int x = 3;
    int y = 7;
    int result;
    result = AlmostProduct(x,y);
    }
```

(a) no, the function names do not match

(b) no, the function call should pass the variables *a* and *b*

(c) no, the function call should pass two values

(d) yes

24. Does the function call match the function definition?

```
int almostProduct(int a,int b)
    {
    return a * (b - 1);
    }

int main()
    {
    int x = 3, y = 7, result;
    result = almostProduct(x,y,x * x);
    }
```

(a) no, there are too many arguments

(b) no, there are too many formal parameters

(c) no, the function names do not match

(d) yes

(e) no, the function call should pass values

Concept: *identifying arguments*

25. Identify the function call arguments in the code below:

```
int compute(int x,int y)
    {
    int a, b;
    a = x + 1;
    b = y - 1;
    return a * b;
    }
int main()
    {
    int j,k;
```

```
        j = 3;
        k = 7;
        result = compute(j + 1,k - 1);
        }
```

(a) the variables *j* and *k*

(b) the values *3* and *7*

(c) the expressions involving variables *j* and *k*

(d) the variables *x* and *y*

26. What are the variables *j* and *k* in the code below:

```
    int compute(int x,int y)
        {
        int a, b;
        a = x + 1;
        b = y - 1;
        return a * b;
        }
    int main()
        {
        int j,k,result;
        j = 3;
        k = 7;
        result = compute(j + 1,k - 1);
        }
```

(a) the formal parameters of a function

(b) function names

(c) local variables defined in the body of the *compute* function

(d) the arguments given in a function call

27. What are the formal parameters of the *compute* function?

```
    int compute(int x,int y)
        {
        int a, b;
        a = x + 1;
        b = y - 1;
        return a * b;
        }
    int main()
        {
        int j, k, result;
        j = 3;
        k = 7;
        result = compute(j + 1,k - 1);
        }
```

(a) *j*, *k*

(b) *x*, *y*

(c) the compute function has no formal parameters

(d) *a*, *b*

28. What is the value of *x* while the function body is being evaluated?

```
        int compute(int x,int y)
            {
            int a,b;
            a = x + 1;
            b = y - 1;
            return a * b;
            }
        int main()
            {
            int j, k, result;
            j = 3;
            k = 7;
            result = compute(j,k);
            }
```

(a) `j`
(b) 3
(c) 7
(d) `k`

29. What is the value of *y* while the function body is being evaluated?

```
        int compute(int x,int y)
            {
            int a,b;
            a = x + 1;
            b = y - 1;
            return a * b;
            }
        int main()
            {
            int j, k, result;
            j = 3;
            k = 7;
            result = compute(j,k);
            }
```

(a) `k`
(b) 7
(c) 3
(d) `y`

30. What are the arguments, formal parameters, and local variables of the *compute* function?

```
        int compute(int x,int y)
            {
            int a,b;
            a = x + 1
            b = y - 1
            return a * b;
            }

        int main()
            {
            int result;
            result = compute(3,7);
            }
```

(a) the compute function has no local variables
(b) x and y, 3 and 7, a and b, respectively
(c) 3 and 7, x and y, a and b, respectively
(d) 3 and 7, a and b, x and y, respectively

31. What is printed by this program:

```
int almostSquare(int x,int y)
    {
    return (x - 1) * (y + 1);
    }

int main()
    {
    printf("%d\n",almostSquare(5,5));
    return 0;
    }
```

(a) 24
(b) 25
(c) `None` because there is no explicit return
(d) nothing is printed because of an error

32. What is printed by this program:

```
int almostSquare(int x,int y)
    {
    return (x - 1) * (y + 1)
    }

int main()
    {
    x = almostSquare(5,5);
    printf("%d\n",x);
    }
```

(a) `None` because there is no explicit return
(b) 24
(c) nothing is printed because of an error
(d) 25

Writing functions in separate files

33. Which of the following statements is true for calling functions from another file?

(a) by convention, it should be redefined
(b) need to be defined in the file, where we are calling the function
(c) it must be defined in another file

34. Which of the following compiling command is true for calling functions from another file?

(a) Use gcc -c to link other .c files
(b) Use gcc -l to link other .c files
(c) Use gcc -o to link other .c files

35. Communication between a calling function and a called function in a same file are provided by following:
    (a) formal parameters and calling arguments
    (b) static variables (assuming static definition for a global variable)
    (c) extern variables
    (d) local variables (assuming local to a function)
36. Communication between a calling function and a called function in a different files are provided by following:
    (a) formal parameters and calling arguments
    (b) local variables (assuming local to a function)
    (c) static variables (assuming static definition for a global variable)
    (d) extern variables

Pass by value and reference

37. The differance in pass by value and reference is the following:
    (a) changes in formal parameters are reflected in the calling function
    (b) reflects changes in local variables from the called function
    (c) all variables defined in called function can be accessed in calling scope
    (d) all variables defined in calling function can be accessed from called function
38. In the functions defined as pass by value, following is not true
    (a) single return value
    (b) can use global variables
    (c) multiple return values
    (d) multiple formal parameters
39. In the functions defined as pass by reference, following is true
    (a) all the answeres
    (b) multiple return values
    (c) multiple formal parameters
    (d) can use global variables

Concept: scope

40. In the main function defined below, what is the output ?

```
#include<stdio.h>
int main()
    {
    int sval  = 1;
        {
        int sval = 2;
        printf("%d",sval);
        }
    }
```

    (a) 1
    (b) garbage value
    (c) 2

41. In the main function defined below, what is the output ?

```
#include<stdio.h>
int main()
    {
    int sval  = 1;
        {
        int sval = 2;
        }
    printf("%d",sval);
    }
```

(a) 1
(b) garbage value
(c) 2

42. In the main function defined below, what is the output ?

```
#include<stdio.h>
int main()
    {
    int sval  = 1;
        {
        int sval = 2;
            {
            int sval = 3;
            }
        }
    printf("%d",sval);
    }
```

(a) 3
(b) 2
(c) 1
(d) garbage value

43. In the main function defined below, what is the output ?

```
#include<stdio.h>
int main()
    {
    int sval  = 1;
        {
        int sval = 2;
            {
            int sval = 3;
            }
        printf("%d",sval);
        }
    }
```

(a) 1
(b) garbage value
(c) 2
(d) 3

44. In the main function defined below, what is the output ?

```
#include<stdio.h>
int main()
    {
    int sval  = 1;
        {
        int sval = 2;
            {
            int sval = 3;
            printf("%d",sval);
            }
        }
    }
```

(a) 2

(b) 3

(c) 1

(d) garbage value

45. What is the output of this program?

```
#include<stdio.h>
int sfun()
    {
    int sval = 10;
    printf("%d ",sval++);
    }

int main()
    {
    for(int i=0; i <=3; i++)
        {
        sfun();
        }
    }
```

46. What is the output of this program?

```
#include<stdio.h>
int sfun()
    {
    static int sval = 10;
    printf("%d ",sval++);
    }

int main()
    {
    for(int i=0; i <=3; i++)
        {
        sfun();
        }
    }
```

47. What is the output of this program?

```
#include<stdio.h>
int sfun()
    {
    static int sval = 10;
    printf("%d ",++sval);
    }

int main()
    {
    for(int i=0; i <=3; i++)
        {
        sfun();
        }
    }
```

48. What is the output of this program?

```
#include<stdio.h>
int sfun()
    {
    int sval = 10;
    printf("%d ",++sval);
    }

int main()
    {
    for(int i=0; i <=3; i++)
        {
        sfun();
        }
    }
```

Writing complete program using functions

49. Write a function to sum the numbers, sum the square of numbers, and sum the cube of the numbers from 1 to x. The user input x is passed as command line argument. The function should have a maximum of three formal parameters HINT: Use pass by reference

50. Write a function to product the numbers, product the square of numbers, and product the cube of the numbers from 1 to x. The user input x is passed as command line arguments. The function should have a maximum of three formal parameters HINT: Use pass by reference

51. Write a function to find the product of two numbers a, b, and sum of two numbers. The user inputs a and b are passed as command line arguments. The function should have a maximum of three formal parameters HINT: Use pass by reference

52. Write a function to find gcdseries for a series of pair of two numbers, starting from a to b (inclusive), where a and b are passed as command line arguments For example:

```
gcdseries (3,5) should find
gcd of 3,4
gcd of 3,5
gcd of 4,5
```

(HINT: gcd of two numbers a, and b can be calculated using the logic: In a loop, update 'b' with 'a' modulus of 'b'; and update 'a' with original of 'b'. Keep doing this in a loop, until 'b' reaches zero. your gcd of (a,b) is 'a'.)

53. Write a function to print prime numbers between two numbers a, b, where a and b are passed as command line arguments (HINT: the isPrime function from cflow quiz document)

54. Write a function to find whether a number (c) lies between two numbers a and b (inclusive) ?

55. Write a function to find the number of even numbers (inclusive) between two numbers a and b, which are passed as command line arguments ?

56. Write a function to find the number of odd numbers (inclusive) between two numbers a and b, which are passed as command line arguments ?

57. Write a function to find fibonacci numbers. For example:

    ```
    fibonacci(1) is 0
    fibonacci(2) is 1
    fibonacci(3) is 2
    fibonacci(4) is 3
    fibonacci(5) is 5
    .........
    .........
    fibonacci(n) = fibonacci(n-1) + fibonacci(n-2)
    ```

    (HINT: Use loop)

58. Write a function to reverse a number. The number is passed as command line argument. For example:

    ```
    reverse(1024) is 4201
    reverse(1000) is 1
    ```

    (HINT: Use loop and expressions involving modulus and division)