

All of the work in this project is my own! I have not left copies of my code in public folders on university computers. I have not given any of this project to others. I will not give any portion of this project to others taking this class. I realize that the penalty for turning in work that is not my own can range from an "F" in the class to dismissal from Trinity University.

Print Title _____ Time Required = _____ Hrs.

Signature _____ (pledged)

Print Title _____ Time Required = _____ Hrs.

Signature _____ (pledged)

Chapter 4 Homework

Individual/Team (1-2 Persons) Assignment
15 Points

Answers To These Questions Must Be Handwritten; No Electronic Solutions Will Be Accepted!
All programming/code questions refers to the C programming language.

When writing program statements, include semicolons where necessary! Remember that C is case sensitive!

- 1] _____ {T/F} The principles of top-down design and structured programming dictate that a program should be divided into a main module and its related modules.
- 2] _____ {T/F} The function definition contains the code for a function.
- 3] _____ {T/F} Function calls that return *void* may not be used as a part of an expression.
- 4] _____ {T/F} The address operator (&) is used to tell the compiler to store data at an address.
- 5] _____ {T/F} Variables defined within a block have global scope.
- 6] _____ The process of dividing a program into functions – which in turn are divided into functions until they consist of only elementary processing that is intrinsically understood and cannot be further subdivided - is known as
 - a. charting
 - b. flow charting
 - c. factoring
 - d. programming
 - e. structuring
- 7] _____ Which of the following statements about function declaration and definition is true?
 - a. The function call is found in the called function.
 - b. The function declaration requires that the parameters be named.
 - c. The function definition is done with a function declaration.
 - d. The function definition contains executable statements that perform the function's task.
 - e. The function definition header concludes with a semicolon (;).
- 8] _____ Which of the following is not a part of a function header?
 - a. name
 - b. parameter list
 - c. return type
 - d. title

9] _____ Which of the following statements about function parameters is true?

- a. Empty parameter lists are declared with the keyword *void*.
- b. If there is only one parameter, the function list parentheses are not required.
- c. In the definition of a function, the parameters are known as actual parameters.
- d. Parameters are separated by semicolons.
- e. The parameters in a function definition are defined in the function's body (local declaration section).

10] _____ Which of the following statements about local variables is false?

- a. A local variable's value may be returned through a *return* statement.
- b. Local variables are defined inside a function.
- c. Local variables cannot be referenced through their identifiers outside the function.
- d. Local variables may be initialized with an initializer.
- e. Local variables' names can be the same as the function's parameter names.

11] _____ To tell the compiler to store data at an address, use the

- a. address operator (&)
- b. array operator ([])
- c. deference operator (#)
- d. indirection operator (*)
- e. pointer operator (^)

12] _____ The function that returns the absolute value of a long integer is

- a. abs
- b. dabs
- c. fabs
- d. labs
- e. tabs

13. Which of the following statements will generate a random number in the range 30-50?

- a. rand (33)
- b. (rand () % 20) + 1
- c. (rand () % 21) + 20
- d. (rand () % 21) + 30
- e. (rand () % 51) + 1

14] _____ Which of the following statements about structure charts is false?

- a. Structure charts are a replacement for flowcharts.
- b. Structure charts are a primary design tool for a program.
- c. Structure charts are used in a structured walk-through to validate the design.
- d. Structure charts can be used to assess the testability of a program.
- e. Structure charts should be created before you start writing a program.

15] Find any errors in the following function definition: (Write something to correct the problem if possible - otherwise just identify the error)

```
void fun (int x, y)
{
    int z;
    ...
    return z;
} // fun
```

16. Find any errors in the following function definition: (Write something to correct the problem if possible - otherwise just identify the error)

```
int fun (int x, y)
{
    int z;
    ...
    return z;
} // fun
```

17. Find any errors in the following function definition: (Write something to correct the problem if possible - otherwise just identify the error)

```
int fun (int x, int y)
{
    ...
    int sun (int t)
    ...
    {
        ...
        return (t + 3);
    }
    ...
    return z;
} // fun
```

18. Find any errors in the following function definition: (Write something to correct the problem if possible - otherwise just identify the error)

```
void fun (int x,)
{
    ...
    return;
} // fun
```

19. Find any errors in the following function declarations/prototypes: (Write something to correct the problem if possible - otherwise just identify the error)

- a. `int sun (int x, y);`
- b. `int sun (int x, int y)`
- c. `void sun (void, void);`
- d. `void sun (x int, y float);`

20. Find any errors in the following function calls: : (Write something to correct the problem if possible - otherwise just identify the error)

- a. `void fun ();`
- b. `fun (void);`
- c. `void fun (int x, int y);`
- d. `fun ();`

21. Evaluate the value of the following expressions:

a. `fabs (9.5) = _____`

b. `fabs (-2.4) = _____`

c. `fabs (-3.4) = _____`

d. `fabs (-7) = _____`

e. `fabs (7) = _____`

22. Evaluate the value of the following expressions:

a. `floor (9.5) = _____`

b. `floor (-2.4) = _____`

c. `floor (-3.4) = _____`

d. `ceil (9.5) = _____`

e. `ceil (-2.4) = _____`

f. `ceil (-3.4) = _____`

23. Evaluate the value of the following expressions when x is 3.5, 3.45, 3.76, 3.234, and 3.4567:

a. `floor (x * 10 + 0.5) / 10 = _____`

b. `floor (x * 100 + 0.5) / 100 = _____`

c. `floor (x * 1000 + 0.5) / 10000 = _____`

24. Define the range of the random numbers generated by the following expressions:

a. `rand () % 10 = _____ to _____`

b. `rand () % 4 = _____ to _____`

c. `rand () % 10 + 1 = _____ to _____`

d. `rand () % 52 = _____ to _____`

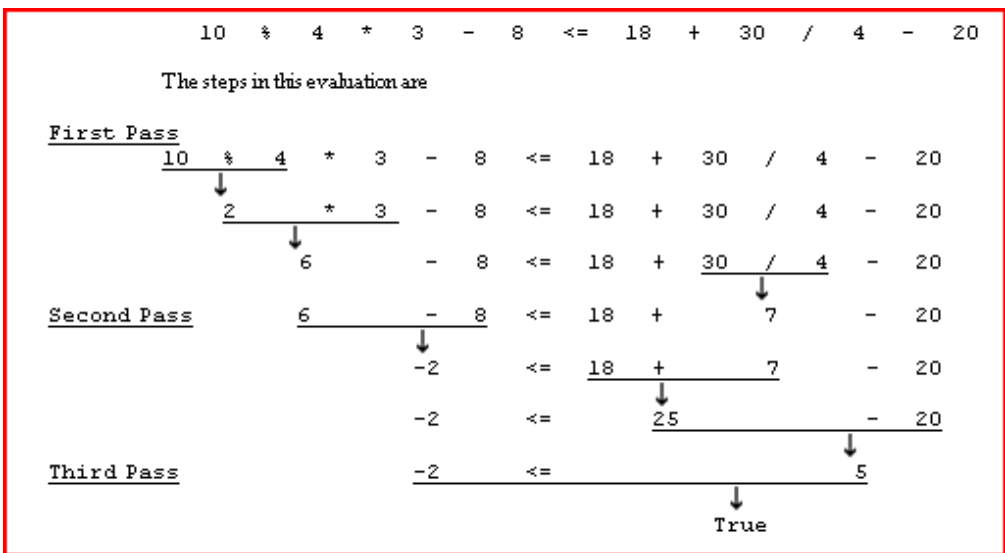
e. `rand () % 2 + 1 = _____ to _____`

f. `rand () % 52 - 5 = _____ to _____`

Priority Table For All Of C

Priority is Highest at Top of Chart and Lowest at Bottom of Chart.
Operations between horizontal lines, above, have the same precedence!

C Operator	Associativity Direction
() Parentheses	Innermost Out
++ (Increment) ,-- (Decrement)	Right to Left
! (Unary Not)	Right to Left
& (Address)	Right to Left
* (De-reference)	Right to Left
& (Address)	Right to Left
(cast type)	Right to Left
- (Unary Minus)	Right to Left
sizeof (returns # bytes for argument)	Right to Left
* (Multiplication)	Left To Right
/ (Division)	Left To Right
% (Remainder)	Left To Right
+ (Addition)	Left To Right
- (Subtraction)	Left To Right
< (Less Than)	Left To Right
> (Greater Than)	Left To Right
<= (Less Than or Equal To)	Left To Right
>= (Greater Than or Equal To)	Left To Right
== (Equal)	Left To Right
= Equal	Left To Right
&& (And)	Left To Right
(Or)	Left To Right
= (Assignment Equal)	Right To Left
+= (Plus Equal Assignment)	Right To Left
-= (Minus Equal Assignment)	Right To Left
*= (Times Equal Assignment)	Right To Left
/= (Divide Equal Assignment)	Right To Left
%= (Modulus Equal Assignment)	Right To Left



Run the following & List the output!

```
printf("(true) ==> %d\n", (true) ); _____  
  
printf("(false) ==> %d\n\n", (false) ); _____
```

This implies that

```
true = 1  
false = 0
```

In reality,

```
true = !0  
false = 0
```

24] Evaluate each of the following as either **true** or **false**. Run the code if you are not sure. If the output from the program would be 0 then put false ; otherwise put true.

```
_____ printf("(true) ==> %d\n", (true) );  
_____ printf("(false) ==> %d\n\n", (false) );  
_____ printf("(false == false) ==> %d\n", (false == false) );  
_____ printf("(false == 0) ==> %d\n", (false == 0) );  
_____ printf("(false == 1) ==> %d\n\n", (false == 1) );  
_____ printf("(true == true) ==> %d\n", (true == true) );  
_____ printf("(true == 1) ==> %d\n", (true == 1) );  
_____ printf("(true == 2) ==> %d\n", (true == 2) );  
_____ printf("(true && true) ==> %d\n", (true && true) );  
_____ printf("(true && false) ==> %d\n", (true && false) );  
_____ printf("(false && true) ==> %d\n", (false && true) );  
_____ printf("(false && false) ==> %d\n", (false && false) );  
_____ printf("(true || true) ==> %d\n", (true || true) );  
_____ printf("(true || false) ==> %d\n", (true || false) );  
_____ printf("(false || true) ==> %d\n", (false || true) );  
_____ printf("(false || false) ==> %d\n\n", (false || false) );  
_____ printf("(! true) ==> %d\n", (! true) );  
_____ printf("(! false) ==> %d\n\n", (! false) );  
_____ printf("%d\n", (4.2 >= 5.0) && (8 == (3 + 5)) );  
_____ printf("%d\n", (4.2 >= 5.0) || (8 == (3 + 5)) );  
_____ (4.2 >= 5.0) && (8 == (3 + 5))  
_____ (4.2 >= 5.0) || (8 == (3 + 5))
```

25] Evaluate each of the following as either **true** or **false**. Run the code if you are not sure. If the output from the program would be 0 then put false ; otherwise put true.

_____ (-2 < 0) && (18 >= 10)

_____ (-2 < 0) || (18 >= 10)

_____ (3 > 5) && (14.1 == 0.0)

_____ (3 > 5) || (14.1 == 0.0)

_____ ! (18 == (10 + 8))

_____ ! (- 4 > 0)

_____ ! 7 == 7

_____ ! -3.0 == 0.0

_____ ! 4.2 > 3.7

_____ ! -18 < -15

_____ ! 13 < 100

_____ 13 <= 100

_____ 13 <= 13

_____ 0.012 > 0.013

_____ -17.32 != -17.32

_____ 4 != 5

_____ X <= Y

_____ Y > X

_____ 'B' > 'A'

_____ 'A' <= 'B'

_____ 'B' > 'a'

_____ 'C' > 'A'

_____ 'a' == 'A'

26] What output is produced from each of the following program fragments? Assume all variables have been suitably declared. Record the output, if any, to the right of the print statement; some will be blank!

a.

```
A = -14.;
B = 0.0;
if (A < B)
    printf ("%f %f\n", A, abs(A));
else
    printf ("%f\n", A * B);
```

b.

```
B = 25;
Count = 0;
Sum = 0;
if (A == B)
    printf ("%d %d\n", A, abs(A)); _____
else
{
    Count = Count + 1;
    Sum = Sum + A + B;
    printf ("%d %d\n", Count, Sum); _____
}
printf ("%d %d\n", Count, Sum); _____
```

c.

```
Temp = 0;
A = 10;
B = 5;
if (A > B)
    printf ("%d %d\n", A, B); _____
else
    Temp = A;
A = B;
B = Temp;
printf ("%d %d\n", A, B); _____
```

d.

```
B = 25;
A = 10;
Count = 0;
Sum = 0;
if (B = 12)
    printf ("%d %d\n", A, abs(A)); _____
else
{
    Count = Count + 1;
    Sum = Sum + A + B;
    printf ("%d %d\n", Count, Sum); _____
}
printf ("%d %d\n", Count, Sum); _____
```

e.

```
int X = 7;
if ( 123 )
    printf ("Now\n"); _____
else
    printf ("is\n"); _____
if ( 0 )
    printf ("the\n"); _____
else
    printf ("time\n"); _____
if ( 1 )
    printf ("to\n"); _____
else
    printf ("get\n"); _____
if ( X = 5 )
    printf ("out\n"); _____
else
    printf ("and\n"); _____
```

27] Identify and correct all errors.

a. `if (Ch != '.')
 {
 CharCount = CharCount + 1;
 printf ('Ch = %c\n', Ch);
 else
 PeriodCount = Period Count + 1;
 }`

b. `if (Age < = 20)
 YoungCount = YoungCount + 1;
 YoungAge = YoungAge + Age;
 else
 {
 OldCount = OldCount + 1;
 OldAge = OldAge + Age;
 }`

c. `if Age <= 20
 {
 YoungCount = YoungCount + 1;
 YoungAge = YoungAge + Age
 }
 else
 OldCount = OldCount + 1;
 OldAge = OldAge + Age`