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 \textit{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)

```c
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)

```c
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)

```c
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)

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

The steps in the evaluation are:

**First Pass**

10 4 * 3 - 8 <= 18 + 30 / 4 - 20

**Second Pass**

6 - 8 <= 18 + 7 - 20

**Third Pass**

-2 <= 25 - 20

True
Run the following & List the output!

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

This implies that

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

In reality,

```c
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.

```c
printf("(true) ==> %d\n", (true));
printf("(false) ==> %d\n", (false));
printf("(false == false) ==> %d\n", (false == false));
printf("(false == 0) ==> %d\n", (false == 0));
printf("(false == 1) ==> %d\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", (false || false));
printf("(! true) ==> %d\n", (! true));
printf("(! false) ==> %d\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.

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>(-2 &lt; 0) &amp;&amp; (18 &gt;= 10)</td>
<td>true</td>
</tr>
<tr>
<td>(-2 &lt; 0)</td>
<td></td>
</tr>
<tr>
<td>(3 &gt; 5) &amp;&amp; (14.1 == 0.0)</td>
<td>false</td>
</tr>
<tr>
<td>(3 &gt; 5)</td>
<td></td>
</tr>
<tr>
<td>! (18 == (10 + 8))</td>
<td>true</td>
</tr>
<tr>
<td>! (-4 &gt; 0)</td>
<td>true</td>
</tr>
<tr>
<td>! 7 == 7</td>
<td>false</td>
</tr>
<tr>
<td>! -3.0 == 0.0</td>
<td>true</td>
</tr>
<tr>
<td>! 4.2 &gt; 3.7</td>
<td>false</td>
</tr>
<tr>
<td>! -18 &lt; -15</td>
<td>true</td>
</tr>
<tr>
<td>! 13 &lt; 100</td>
<td>false</td>
</tr>
<tr>
<td>! 13 &lt;= 100</td>
<td>true</td>
</tr>
<tr>
<td>! 13 &lt;= 13</td>
<td>true</td>
</tr>
<tr>
<td>! 0.012 &gt; 0.013</td>
<td>false</td>
</tr>
<tr>
<td>! -17.32 != -17.32</td>
<td>true</td>
</tr>
<tr>
<td>! 4 != 5</td>
<td>true</td>
</tr>
<tr>
<td>! X &lt;= Y</td>
<td>true</td>
</tr>
<tr>
<td>! Y &gt; X</td>
<td>true</td>
</tr>
<tr>
<td>'B' &gt; 'A'</td>
<td>true</td>
</tr>
<tr>
<td>'A' &lt;= 'B'</td>
<td>true</td>
</tr>
<tr>
<td>'B' &gt; 'a'</td>
<td>true</td>
</tr>
<tr>
<td>'C' &gt; 'A'</td>
<td>true</td>
</tr>
<tr>
<td>'a' == 'A'</td>
<td>true</td>
</tr>
</tbody>
</table>

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!

```c
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
", A, abs(A));
else
    { 
    Count = Count + 1;
    Sum = Sum + A + B;
    printf("%d  %d
", Count, Sum);
    }
printf("%d  %d
", Count, Sum);

c.
Temp = 0;
A = 10;
B = 5;
if (A > B)
    printf("%d  %d
", A, B);
else
    Temp = A;
    A = B;
    B = Temp;
printf("%d  %d
", A, B);

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