Part I: Multiple Choice (24 points)

Answer all of the following questions. READ EACH QUESTION CAREFULLY. Fill the correct bubble on your mark-sense sheet. Each correct question is worth 2 points. Choose the one BEST answer for each question. Assume that all given C code is syntactically correct unless a possibility to the contrary is suggested in the question.

Remember not to devote too much time to any single question, and good luck!

1. Which one of the following is a difference between a variable and a `#define` constant?

A. A variable's name cannot contain underscores, while a constant's can.
B. A variable can be of any type, while a constant can only be of type double.
C. Only variables can be passed as arguments to functions.
D. The value of a variable can be changed after initialization, but the value of a `#define` constant cannot.
E. Variables can be used as operands in arithmetic expressions, but constants cannot.

2. Suppose that you are given two C library functions that compute the sine and cosine and they are declared as follows:

```
double sin(double angle);
double cos(double angle);
```

Which of the following arithmetic expressions in C is equivalent to the mathematical formula

$$\frac{\mathbf{\textit{sin}^2(x) + \textit{cos}^2(x)}}{\mathbf{\textit{sin}(x)\ \textit{cos}(x)}}$$

A. `sin(x) * sin(x) + cos(x) * cos(x) / sin(x) * cos(x)`
B. `sin(x) * sin(x) + cos(x) * cos(x) / (sin(x) * cos(x))`
C. `(sin(x) * sin(x) + cos(x) * cos(x)) / sin(x) * cos(x)`
D. `(sin(x) * sin(x) + cos(x) * cos(x)) / (sin(x) * cos(x))`
E. none of the above

3. Which programming term(s) accurately describes what the following statement does?

```
int temperature = 57;
```

I. initialization
II. declaration
III. type conversion
IV. function call

A. I only
B. I and II only
C. II and III only
D. II, III, and IV only
E. I, II, and IV only

4. **When the following fragment of code is executed, what is the output?**

```
int x = 5;
if (x >= 5) {
    printf("x is greater than or equal to 5.\n");
} else if (x <= 5) {
    printf("x is less than or equal to 5.\n");
} else {
    printf("x is equal to 5.\n");
}
```

A. `x is greater than or equal to 5.`
`x is equal to 5.`
B. `x is less than or equal to 5.`
`x is equal to 5.`
C. `x is greater than or equal to 5.`
D. `x is less than or equal to 5.`
E. `x is greater than or equal to 5.`
`x is less than or equal to 5.`

5. **According to rules of arithmetic evaluation in C, what is the result of evaluating the following expression?**

```
2 * 15 - 15 % (5 * 2)
```

A. 0
B. 20
C. 25
D. 28
E. 29

6. **When the following (poorly formatted but syntactically correct) fragment of code is executed, what is the output?**

```
int x = 0, y = 0, a = 2, b = 1;
double z = -4.0;

if (x == 0)
if (y != 0)
z = b - a;
else
z = a + b;

printf("%f", z);
```

A. 3.000000
B. -4.000000
C. -1.000000
D. 2.00000
E. 0.000000

7. **Given the function definitions below, what will be the output produced by the `printf` in `main`?**

```
int crunch(int a, int b, int c) {
    return (a * b / c);
}

int main(void) {
    double y;
    y = crunch(6, 7, 8);
    printf("%f", y);
    return (0);
}
```

A. `0`
B. `0.000000`
C. `5`
D. `5.000000`
E. `5.250000`

8. **Which of the following is NOT considered good programming practice?**

A. breaking up your program into a set of functions that each does a small piece of the work
B. using nested if-else statements
C. using conditionals in which the expression in the parentheses involves an assignment
D. using conditionals in which the expression in the parentheses involves two or more Boolean operators
E. using explicit type conversions (casting)

9. **All but one of the following are legal identifier names for C variables. Which one is not a legal variable identifier?**

A. `xyzzy_10`
B. `4th`
C. `AsillyIDEnTiFier`
D. `How_about_THIS1`
E. `SOMETHING_BIG`

10. **When the C arithmetic expression**

```
x + y * z
```

is evaluated, which of the following is true?

A. **`x + y`** is evaluated first because **`+`** and **`*`** are left-associative
B. **`y * z`** is evaluated first because **`+`** and **`*`** are right-associative
C. **`x + y`** is evaluated first because **`+`** has higher precedence than **`*`**
D. **`y * z`** is evaluated first because **`*`** has higher precedence than **`+`**
E. Since there are no parentheses to force the order of evaluation, it is not possible to tell in advance which subexpression will be evaluated first.

11. In the function definition

```
void mumble(void)
{
   ...
}
```

what is the meaning of the first "`void`" **(the one to the *left* of the function name** `mumble`**)?**

A. The function body must be empty, i.e., have no statements in it.
B. The function does not have any parameters.
C. The function may have parameters, but it ignores their value(s).
D. The function does not produce a result value.
E. The function may return a value, but the returned value will be ignored by the caller.

12. Which one of the following statements is *exactly* equivalent to the following assignment? (All variables and values are `int`s.)

```
w = 5 * x % 2 - (3 + 7);
```

A. `w = 5 * (x % 2) - (3 + 7);`
B. `w = (5 * x % 2) - (3 + 7);`
C. `w = (5 * x) % (2 - 3) + 7;`
D. `w = 5 * x % (2 - 3 + 7);`
E. `w = 5 * x % 2 - 3 + 7;`

Part II: Programming Questions (26 points)

Write C code for the following two problems. For both programs, input checking is not required; you should assume that the user will provide sensible input values.

13. (10 points) Jane is making big digital signs for billboards. Each sign will have several lines of text. Given the width of a billboard (`bbwidth`) and the width of a character (`charwidth`), both as `double`s, Jane must compute how many characters will fit on one line of text on that billboard and how much space is left over. Write a program that reads `bbwidth` and `charwidth`, then calculates and prints the following in order:

(1) `numchars`, the number of characters that will fit on one line
(2) `spaceleft`, the amount of space that will be left over on each line

Notice that we have not specified the units of measurement (feet, centimeters, etc.) and that this information is not needed, as long as the same units are used for both input values.

example run (user input is underlined):

```
    bbwidth? 13.35
    charwidth? 2.0

    numchars = 6
    spaceleft = 1.350000
```

```
#include <stdio.h>

int main(void) {
    /* add variable declarations for numchars and spaceleft below */
    double bbwidth, charwidth;



    /* read billboard and character widths */










    /* compute and print numchars and spaceleft */















    return (0);
}
```

14. (16 points) You own an apartment building and have to collect rent from the tenants. All rent is officially due on the first day of the month, but there is a grace period of 5 days. If the rent comes in on the first, second, third, or fourth day, you are happy and want to thank the tenant. If it comes in on the fifth day, you are relieved and want to warn the tenant to be careful next time. If it comes in after the fifth day of the month, you charge the tenant a late fee that is the larger of 5% of the rent or $20.00.

The program **on the next page** already contains code to read

(1) an `int` giving the day of the month (between 1 and 31) that the tenant paid the rent and
(2) a `double` that gives the amount of the rent.

Complete the program on the next page so that it prints

```
Thank you.
```

if the tenant pays on days one through four, or prints

```
Rent received, but be careful not to be late!
```

if the tenant pays on day five, or prints

```
Rent overdue.  Late fee is $__.__.
```

after day five. (The actual amount of the late fee should be printed in place of the underscores.)

(Do not write your answer here. See next page.)

```
#include <stdio.h>

int main(void) {
    int date_paid;  /* day on which rent was paid */
    double rent;    /* tenant’s monthly rent */

    /* declare any additional variables you need below */









    /* ask for, then read date and rent amount */
    printf("Enter date paid and monthly rent: ");
    scanf("%d %lf", &date_paid, &rent);

    /* print appropriate message, including late fee if any
    /* write your code below */



























    return(0);
}
```