1. In the program below, label one example of each of the following: comment, preprocessor directive, function call, operator, assignment statement, variable declaration

```
// Program to calcuate miles per gallon
#include <iostream>
using namespace std;

int main() {
      double miles, gallons, MPG;

      cout << "Enter the number of miles: ";
      cin >> miles;

      cout << "Enter the number of gallons: ";
      cin >> gallons;

      MPG = miles / gallons;

      cout << "Your mileage was " << MPG << " miles per gallon." << endl;

      return 0;
} // end main
```

2. The `cout` object is *stream* object that works with a stream of data to display on the console output. The console will display the stream all on one line unless it is told to go to the next line by a *newline character*. The newline character can be inserted into the stream in two ways:

Ways to insert a newline character	Examples
Inserting the stream manipulator `endl`	`cout << "string literal" << endl;` `cout << "line 1" << endl << "line 2" << endl << "line 3";`
Inserting the newline escape sequence, `\n`	`cout << "string literal\n";` `cout << "line 1\nline 2\nline 3";`

Predict the output of the below partial program.

```
cout << "my name";
cout << "is " << endl <<"John\n";
cout << "Doe\n" << "endl";
cout << "and I'm fine";
```

Output

3. An *identifier* is a programmer-defined name for some part of a program: variables, functions, etc.

- The first character of an identifier must be an alphabetic character or and underscore (_),
- After the first character you may use alphabetic characters, numbers, or underscore characters.

Uppercase and lowercase characters are distinct. Circle the syntactically correct identifiers below:

phone_#	timeOfDate	8ofHearts
distanceToTravel	RATE_OF_PAY	_____494_____

4. Complete the following table.

	Decimal (Base 10)	**Binary (Base 2)**	**Hexadecimal (Base 16)**
Number of digits:	10		
Digits:	0, 1, 2, 3, 4, 5, 6, 7, 8, 9		
Counting:	0		
	1		
	2		
	3		
	4		
	5		
	6		
	7		
	8		
	9		
	10		
	11		
	12		
	13		
	14		
	15		
	16		
	17		
	18		
	19		
	20		
	21		

5. Convert 173_{10} to a binary (base 2) value.

6. ASCII Character Representations are below. What do you notice about the following group of characters:

0 NUL	16 DLE	32	48 0	64 @	80 P	96 `	112 p
1 SOH	17 DC1	33 !	49 1	65 A	81 Q	97 a	113 q
2 STX	18 DC2	34 "	50 2	66 B	82 R	98 b	114 r
3 ETX	19 DC3	35 #	51 3	67 C	83 S	99 c	115 s
4 EOT	20 DC4	36 $	52 4	68 D	84 T	100 d	116 t
5 ENQ	21 NAK	37 %	53 5	69 E	85 U	101 e	117 u
6 ACK	22 SYN	38 &	54 6	70 F	86 V	102 f	118 v
7 BEL	23 ETB	39 '	55 7	71 G	87 W	103 g	119 w
8 BS	24 CAN	40 (	56 8	72 H	88 X	104 h	120 x
9 HT	25 EM	41)	57 9	73 I	89 Y	105 i	121 y
10 LF	26 SUB	42 *	58 :	74 J	90 Z	106 j	122 z
11 VT	27 ESC	43 +	59 ;	75 K	91 [	107 k	123 {
12 FF	28 FS	44 ,	60 <	76 L	92 \	108 l	124 |
13 CR	29 GS	45 -	61 =	77 M	93]	109 m	125 }
14 SO	30 RS	46 .	62 >	78 N	94 ^	110 n	126 ~
15 SI	31 US	47 /	63 ?	79 O	95 _	111 o	127 DEL

a) Upper-case letters:

b) lower-case letters:

c) digits:

d) nonprintable characters:

7. Explain the output of the following program.

```
// Program containing some confusing stuff
#include <iostream>
using namespace std;

int main() {
     int myInt;
     double myDouble;
     char ch, ch2;

     myInt = 022;
     cout << "myInt: " << myInt << endl << endl;

     ch = 'A';
     cout << "ch: " << ch << endl << endl;
     ch2 = ch+1;
     cout << "ch+1: " << ch+1 << "  ch2: " << ch2 << endl << endl;
     ch2 = ch+32;
     cout << "ch+32: " << ch+32 << "  ch2: " << ch2 << endl << endl;
     ch = '\n';
     cout << "static_cast<int>(ch): " << static_cast<int>(ch)<<endl<< endl;
     ch = 8;
     cout << "What is this line?" << ch << ch << ch <<"abcd"<<endl << endl;

     myDouble = 7.9;
     myInt = myDouble;
     cout << "myInt: " << myInt << endl << endl;
     myDouble = 3.14e19;
     myInt = myDouble;
     cout << "myInt: " << myInt << endl;

} // end main
```

```
myInt: 18

ch: A

ch+1: 66  ch2: B

ch+32: 97  ch2: a

static_cast<int>(ch): 10

What is this liabcd

myInt: 7

myInt: -2147483648
```

Hint: The above program compiled fine (no errors), but it did give two warning messages:

```
1. warning C4244: '=' : conversion from 'double' to 'int', possible loss of data
2. warning C4244: '=' : conversion from 'double' to 'int', possible loss of data
```

MORALE: Pay attention to 'Warnings'!!! They usually indicate that you are doing something wrong!

8. When writing a C++ program, you should use good *style* to make your code more readable for human beings. Some of general conventions of good style are:

- use meaningful variable names with good style, i.e., useCamelCase (or use_underscores) (I like to declare them one per line with a following comment if necessary)
- use meaningful named constants (e.g, PI, STATE_SALES_TAX) where appropriate with good style (ALL_CAPS_AND_UNDERSCORES). Put your global constants where they can be found and changed easily in future versions of your program, e.g, after your initial comments describing the program and before your main function definition.
- use comments at the start of the program, before each function, and before any especially difficult section of code to understand (I like to label the closing set bracket, '}', with some indication of what's being closed)
- place the `main` function near the top of the program with user-defined functions below it
- use *white space* (spaces, indentation, blank lines) to make you program more readable by:
  - aligning the opening set/curlly brace, '{', with the corresponding closing '}' one (I like to put the '{' on the same line with the programming construct (e.g., main function definition) with the closing '}' aligned with the start of the construct)
  - indent all the lines inside a set of of braces
  - use blank lines to separate logical units of the code, e.g., between variable declarations and executable statements

```
// Program to calculate miles per gallon

#include <iostream>
using namespace std;

int main() {
      double miles;     // variable receiving the number of miles driven input
      double gallons;   // variable receiving the gallons of fuel consumed input
      double MPG;       // variable receiving the calculated Miles Per Gallon

      // Interactively gather the miles driven and number of gallons used
      cout << "Enter the number of miles: ";
      cin >> miles;

      cout << "Enter the number of gallons: ";
      cin >> gallons;

      // Calculate the miles per gallon
      MPG = miles / gallons;

      // Display results to the console
      cout << "Your mileage was " << MPG << " miles per gallon." << endl;

      return 0;         // send "successful execution" signal to OS
} // end main
```

NOTE: the above program arguably has too many comments. Which comments are unnecessary?