A. Simple C program

1. Refer to pi.c
   a. In what line is TEST_VALS defined?
      In line 2, it is defined as an array consisting of -1, 0, 1, 2, 10, 100

   b. In what line is TEST_VALS used?
      TEST_VALS is first used in line 20, when the temporary variable vals[] is set equal to it

   c. What characters mark... comments?
      // or /* and *//
   d. characters strings?
      ""
   e. the start and end of a block of code?
      { }
   f. How many functions are defined in the code? List their names
      Functions:
         estimatePi(int terms)   return type - double
         testEstimatePi()   return type - void
         interact()   return type - void
         main()   return type - void (?)
      Total # of functions - 4

   g. What is the return type for estimatePi()?
      double

   h. What are the input parameters for estimatePi()?
      int terms

   i. What keyword marks a function with no return value?
      void

   j. What function actually prints text?
      interact() and testEstimatePi()

   k. Languages structures that control the order of code are called control structures (e.g. if, for, while).
      Which control structures look like Java or C#?
      the if and else blocks look like Java, as well as the for and while loops

2. Describe how you would change this program to add a test for 1000 terms. For each change specify the line number and describe what the change should be.
   I would change line 2, which defines the variable TEST_VALS, to { -1, 0, 1, 2, 10, 100, 1000}. I would add the element 1000 to that int array.
   This would allow for the number 1000 to be tested as an input for the testEstimtatePi() function.

B. Preprocessor Directives

1. Use the instructions above (refer to notes) to write C directives that...

   a. include math.h
      #include <math.h>

   b. Define PI to be 3.14159
      #define PI 3.14159

   c. Define SUM3 to take 3 values and return their sum
      #define SUM3(A,B,C) ((A) + (B) + (C))

2. Which approach (function or macro)...
    a. evaluates arguments before passing them to the body?
        function
    b. uses extra time and memory?
        function
    c. is likely to be faster?
        macro

C. Checking for Errors

1. Refer to error.c
    a. Line 10 would print 97 and 101
    b. You cannot reference an index out of bounds or a negative index in Java or C#
    c. if (index >= size || index < 0)
    d. It would take 10000 instructions b/c 10 * 1000 is 10000

2. C would not check for these errors because it takes more energy, and isn't as efficient. It's better for the programmer to implement these checks as he/she sees fit

3. I would prefer a language that checks because it simplifies my job. However, if I was concerned with speed and efficiency, I would prefer a language that requires more work on my end but is faster.

D. Text Input and Output (I/O)

1. In C, a common and flexible way to print output is with printf(),
   which prints to the standard output (also called stdout), usually the terminal window.
   The first argument specifies how to print later arguments, and is called a format string.
   For each later argument, the format string has a conversion specification.
   Use the C code and its output above to answer the questions below.
    a. A tab is "\t"
    b. A newline is "\n"
    c. A conversion specification is indicated by "%"
    d. A character is specificed by "%c"
    e. A string is specificed by "%s"
    f. A deciminal integer with at least 4 characters is specificed "%7.3f"
    g. "%%" produces an actual percent sign, it acts as an escape character.
2. The conversion specification for floating point numbers can contain 1 or 2 integers separated by a period.
    a. The 1st integer specifices how many total characters should be printed from the number
    b. The 2nd integer specifices how many characters after the decimal should be printed
    c. A leading zero indicates that 0's should be printed instead of spaces
3. printf() returns the number of characters that are printed
    a. In line 03, printf() would return 6
    b. In line 08, printf() would return 8
4. In C, a common and flexible way to read input is with scanf(),
   which reads from the standard input (also called stdin), usually the terminal window.
   The first argument is a format string that specifies how to scan values for later arguments.
   The information for each argument is called a conversion specification.
   Use the C code and expected input above to answer the questions below
    a. "%c" specifies a character should be read
    b. "%s" specifies a string should be read
    c. "&" usually comes before the later arguments
5. scanf() returns the number of values (not characters) that are scanned.
    a. In line 03, scanf() returns 1
    b. In line 05, scanf() returns 1
6. C has other I/O functions that are similar to printf() and scanf()
   but use a file or a string instead of stdout or stdin.

a. fprintf() would allow me to print to a file.
b. The s in sprintf() stands for string, as you write to a string
c. sscanf() would allow you to read from a string
d. Yes, you can, but it might cause problems

7. We don’t always know the exact format of input that a program should read. Explain why it is often easier to scan a full line as a string, and then scan that string.

Because you are guaranteed that the input will at the very least be a string, and then you can parse and work with that string

8. Refer to stat.c