A (Data Comms)

Draw the waveform that would be produced for the ASCII character C (67₁₀) by an analogue transmission device using the adjacent QAM coding scheme. The character is to be transmitted with even parity, but no start or stop bits. A 0° phase shift at the start of the trace. You should indicate clearly where the signalling elements occur in the transmitted character, and what bits they encode.

NOTE: parity bit should be added at the end of the character

You may assume that the tail end of the waveform associated with the previous character looks like this:

B (Data Comms)

Draw a phase diagram showing how a five-bit code could be mapped onto an analogue carrier wave using QAM with four amplitude levels. Organise the codes so that a single-position error in the phase and amplitude of a signalling element only causes a single-bit error in the decoded bit sequence.
C (Combinatorial Logic)
A transparent wheel on a shaft is to be used to indicate the rotational position of the shaft. The wheel is painted with a pattern (the grey region on the adjacent diagram) so that a series of eight optical sensors arranged uniformly around the wheel (the little black circles on the adjacent diagram) can be used to detect its position, with a precision of one eighth of a rotation. A sensor emits a logic True output when the hole in the pattern allows the sensor to detect light shining through from the other side of the wheel, and a logic False otherwise. The pattern has been carefully painted so that no two sensors ever emit a True output simultaneously. Design a logic circuit that converts the Boolean outputs from the eight sensors into a three-bit binary number. Design in terms of idealised gates, not commercially available chips.

D (Combinatorial Logic)
Draw a diagram of the wheel with paint pattern revised so that only three sensors are required, and no encoding circuitry is required. Show the positions of the sensors on your diagram.

E (Assembler Programming)
Write a program that fills memory locations 0060h to 06Fh with the values 0 to 15, putting 0 into location 60, 1 into 61.

F (Assembler Programming)
Using a loop, write a block copy subroutine that takes three parameters:
- a start address
- a destination address
- a byte-count
and moves the specified number of bytes in internal memory from the source to the destination addressed. The parameters should be passed in registers. Your solution should show the subroutine being called from a main program.

G (Assembler Programming)
The position of a rotating disk is indicated by the activation of one of eight light sensors, as specified in question C, above. Write a routine to take the place of the logic circuit you designed in the answer to question C. That is, the routine will translate the single-bit inputs from the eight sensors into a three bit binary number.

Your routine should assume that the data from the light sensors is present on Port 1, with each bit indicating the value of one of the eight sensors. Only one of the eight bits will be high at any one time. The routine will return a 3-bit value in the accumulator indicating which light sensor is active. You can use a sequence of comparisons rather than a lookup table and a look. This makes the program longer but very simple.

NOTE:
While each of these programs is quite short, you must contain both an overall block comment before each subroutine or main block, and end-of-line comments.
comment should be sufficient to explain what your program or subroutine does, and should also indicate any pre- and post-conditions, such as the locations of any parameters and returned values. The end-of-line comment explains the effect of each line. Comments are included in a source file by preceding them with a semicolon - the remainder of the current line will be ignored by the assembler.

When running the simulator, use the ALT/PrintScrn key combination to copy a picture of the current window to the clipboard. Each snapshot can then be pasted into Word or Wordpad for adding explanatory comments and printing (notepad will not work).

For each of the parts E, F, and G, submit an error-free assembler output listing (not the unassembled 8051 source file) and several screen snapshots of your program being run on the simulator. At a minimum, these screen shots should show the state of the simulator (1) after your program has been loaded but before it has been run, and (2) when your program has run to completion.