Problem 1  (Boolean algebra and pointers on the Handy Board)
Design and write a program on the Handy Board that first encrypts and then decrypts 8-bit secure communication. This is not cypher encryption, but rather encryption using an XOR key. The idea is quite simple: Any binary number which is bit-wise XOR-ed by other binary number (called a key) produces an encrypted number. XOR-ing the encrypted number with the same key again, reproduces the original number, e.g. The character ‘A’ can be represented as the decimal index number 65 on the ASCII table. Take 65 and XOR it with the given decimal key 84. The result is 21. 21 is the encrypted byte. Applying the same key to 21, i.e. 21 XOR 84, returns the original value 65.

Your program should display the message “Hello 007” on the HB’s LCD. Pressing the start button should result in the HB XOR-encoding the message character by character and displaying the encoded message (as integer data) on the LCD. Pressing the stop button should result in the HB decrypting the message and displaying the decrypted version on the LCD. Use the hexadecimal number 6C as the 8-bit encryption key.

You may not use global variables. Make use of pointers (or passing by reference) whenever possible to pass arguments to and return values from functions.

64- and 128-bit keys are used in a similar fashion to facilitate secure internet communication.

Problem 2  (Multidimensional arrays and pointers on the workstation)
In the digital era a weather satellite takes digital images of a storm in Florida. Each image is a 2D array and is saved inside a 3D array, with the 3rd dimension indicating the time order in which the images were taken.

a) Design and write a program that accepts a 3x3 2D integer array from a user (simulating a very low resolution camera). The program then copies the 2D array (image) into a 3D array in the sequence that the images are received. As time progresses the program overwrites the original image (2D array) with a new entry, saves it into the 3D array and so on, until the 3D array is full. Let your 3D storage array be of dimension 3x3x3. Write separate functions for receiving user input and for copying the 2D array into the 3D array.

b) When the 3D array is full, retrieve each image from it and display the contents of each image as a 3x3 block of numbers on the screen. A separate function should receive the single images from the 3D array and display them on the screen.

Do not use global variables or structures and only pass variables by reference as arguments to any and ALL functions in this problem.

Hint: Creating pointers to arrays within arrays will simplify you task immensely. You could e.g. create a pointer to each image within the 3D storage array. Recall the lecture of Friday 16 March.