# tut7 midterm test questions

1. Let p be an int* variable, a pointer to int. Once p is initialized with a valid address, p can be used as if it is the name of an array. For example, we can use p[5] for the 5th entry of the array p[]. If p is a pointer pointing to an array of pointers to int, then we are able to use p[5][4] to access the entry in row 5 column 4 of a 2D array.
Write a program that will prompt for a number n. Create a 2D array of size nxn dynamically, i.e, by calling the new operator to allocate the memory space needed. Place the value 0, 1, 2, … in row major order into the 2D array, i.e., from row 0 column 0 to row 0 column n-1, then to row 1 … , and finally from row n-1 column 0 to row n-1 column n-1, print out the whole 2D array, with each row being printed on a separate line, and then release all the dynamic memory that have been allocated before exit.

2. **Write a complete program** to create an STL vector of integers, generate **randomly** 10 integers to be pushed into the vector, print the content of the whole vector, prompt the user for a number to be checked, then scan the vector and print out "true" or "false" as to whether the number is found.

3. Implement a version of vector called myvec in a **template class** with **T** as type parameter. Below is the myvec ADT.

| | |
|---|---|
| myvec() | // The default constructor which creates an internal array of **capacity** 1. |
| ~myvec() | // Destructor |
| bool isEmpty() | // Return true if the vector is empty |
| int size() | // Return the number of the entries occupied. |
| **T** at(int *index*) | // Return the element found at *index* (0-based) |
| void push_back(**T** *value*) | // Enter the *value* into the last entry |
| bool pop_back() | // Discard the last entry. Return true if the operation is successful |
| int capacity() | // Return the capacity of the array |

Internally a dynamic array is used. When it is full, the array should have its size doubled. **When the size is less than ¼ of the capacity of the array, the array should shrink to ½ of its capacity.**

Since it is a template class, you are required to provide 2 files only, with the first file which is **myvec.h** containing all the codes for the class myvec, and the second file which is **testmyvec.cpp**, a test driver used to show the working of all the methods implemented. Note that your test driver should demonstrate that the vector can be extended or shrunk successfully as well.