Institute for Advancing Intelligence, TCG CREST
(TCG Centres for Research and Education in Science and Technology)

Introduction to Programming and Data Structures
Ph.D. Coursework: First year, First Semester (Session: 2024-25)
Assignment #05

Full Marks: 100 | Instructor: Dr. Laltu Sardar
Clarification Deadline: **2024-Oct-31** | Submission Deadline: **2024-Nov-03**

Instructions:

1. This assignment must be submitted before the next class.
2. You may seek help from anyone or anything if you are stuck. However, do not simply copy and paste. If needed, first understand the solution and then type it out yourself.
3. Keep the C code with you. You will need to give a 5-10 minute demo to explain that you understand what you have implemented. Submit a printed copy to me before giving the demo.

Problem #0501: Stack using Array

Implement stack operation on integer data using **array**. First take a stack structure `struct stack` as `Stack`. Then implement the followings

- `Stack` ← `initialize_stack(int size)`
- `bool` ← `push(Stack stack, int element)`
- `int` ← `pop(Stack stack)`
- `int` ← `peek(Stack stack)`
- `bool` ← `is_empty(Stack stack)`
- `bool` ← `is_full(Stack stack)`

*Function definitions: As discussed in the class

Problem #0502: Stack using Linked List

Implement stack operation on integer data using **Linked list**

Problem #0503: Queue using Array

Implement queue operations on integer data using **array**. First, take a queue structure `struct queue` as `Queue`. Then implement the followings:

- `Queue` ← `initialize_queue(int size)`
- `bool` ← `enqueue(Queue queue, int element)`
- `int` ← `dequeue(Queue queue)`
- `int` ← `front(Queue queue)`
- `bool` ← `is_empty(Queue queue)`
- `bool` ← `is_full(Queue queue)`

Problem #0504: Queue using Linked List

Implement Queue operations on integer data using **Linked list**

Problem #0505: Making Library

Our target is to make a library of important and frequently used utility functions for memory management, file handling, etc.

- Implement at least the following functions, with proper error handling, in the library:
  - `void* allocate_1d_array(int length, int unit_size)`
  - `void** allocate_2d_array(int rows, int cols, int unit_size)`
    – This function should dynamically allocate a 2D array of given size memory with the specified number of rows and columns and return a pointer to the array.
  - `void free_2d_array(void** array, int rows)`
    – This function should free the memory allocated for a 2D array.
  - `FILE* open_file(const char* filename, const char* mode)`
    – This function should open a file with the specified filename and mode (e.g., `"r"` for read, `"w"` for write) and return a file pointer. If the file fails to open, it should print an error message and return `NULL`.
  - `void close_file(FILE* file)`
    – This function should close the given file pointer and print an error message if the file could not be closed.
  - `void sort(int* array, int length)`
    – This sort an integer array
  - `void sort(float* array, int length)`
    – This sort an float array
  - If you find any other frequently used function, then write them.
- Create a suitable menu to test the above functions

Note: The above codes (#0501 to #0505) will be used in the future assignments