CSE 240
Homework 2 – Programming with C

1. What This Assignment Is About:

- Structures
- Functions
- Arrays of Primitive Values
- Arrays of Structs
- Recursion
- for and if Statements
- Insertion Sort

2. Use the following Guidelines

- Give identifiers semantic meaning and make them easy to read (examples num_students, gross_pay, etc).
- Use lower case word for all identifiers (variables, functions, objects). Separate words with underscore character
- Use tabs or spaces to indent code within blocks (code surrounded by braces). This includes structures, functions, and code associated with ifs, switches and loops. Be consistent with the number of spaces or tabs that you use to indent.
- Use white space to make your program more readable.

For each file in your assignment, provide a heading (in comments) which includes:

- **The assignment number.**
- **Its author (your name).**
- **A description of what this program is doing.**

3. Part 1. Primitive Types, Searching, Recursion (35 points).

a) In the file **homework_part_1.c** (available on Canvas), add header comments and function comments

b) Implement the function **initialize_array** that receives two parameters: an array of integers and the array size. Use a for loop and an if statement to put 0s in the odd positions of the array and 5s in the even positions. You must use pointers to work with the array. **Hint: review pointers as parameters.**

c) Implement the function **print_array** that receives as parameters an array of integers and the array size. Use a for statements to print all the elements in the array. You must use pointers to work with the array. **Hint: review pointers as parameters.**

d) Implement the function **insertion_sort** that receives as parameters an array of integers and the array size, and order the array's elements in descending order. Implement Insertion Sort algorithm. It should be Insertion Sort, not Selection Sort, not Quick Sort, etc.

e) Implement the recursive function that calculates and returns the **factorial** of a number. The function receives the number (integer number) as parameter

f) Compile and run the code for **homework_part_1.c**

Any changes made to the main method of the file will be penalized unless otherwise instructed

Grading Criteria for the part 1

01 pts: file contains header information
01 pts: adequate comment to explain every function
01 pts: consistent indentation and spacing
08 pts: insertion_sort
08 pts: print_array
08 pts: initialize_array
08 pts: factorial

4. Part 2 Structs and Arrays (65 points).

In this assignment, we will be making a program to populate a game board with pieces and to move those pieces around on the board.

Use the file **homework_part_2.c** (available on Canvas). You will need to add header comments to each file along with comments for all the methods you will be implementing. You will need to update the condition in the main loops while statement and fill out the procedures below the main procedure.

Any changes made to the main method of the file will be penalized unless otherwise instructed.

Step 1.

First, you need to create a structure **game_piece.** It should contain the variable, label (char [30]). In addition, the following functions should be defined.

Function	Description
void game_piece_init_default(struct game_piece* piece)	Assign the default string "---" to the game piece's label.
void game_piece_init(struct game_piece* piece, char* new_label)	Assign the game piece's label with the new_label provided.
char* game_piece_get_label(struct game_piece* piece)	Returns the piece's label.
char* game_piece_to_string(struct game_piece* piece)	Constructs a C string of length 3 from the piece's label (Note: this length does not include the null character). If the label is shorter than length 3, then the new string should be the label with spaces appended to make it the correct length. If the label is longer than 3, then use the first 3 characters of the label.

Step 2.

You will be creating a structure called **game_board** in the same code file. The structure will contain a 2-dimensional array called "board" of **game_piece** type and 2 ints, rows and columns. Define the following functions:

Function	Description
void game_board_init(struct game_board* game_board, int rows, int cols)	Instantiates the 2-dimensional array "board" to the size "rows" by "columns" specified by the parameters, then sets the game_board's rows and columns values. Then it initializes each game_piece element of this array using the **game_piece_init_default** function. So, each piece will have the default value for its label.

int game_board_is_space_valid(struct game_board* game_board, int row, int col)	The function checks if the parameters row and col are valid. If at least one of the parameters "row" or "col" is less than 0 or larger than the last index of the array (note that the number of rows and columns can be different), then it return 0 (false). Otherwise it returns 1 (true).
int game_board_add_piece(struct game_board* game_board, struct game_piece* piece, int row, int col)	This function should validate that the space specified by row and col is valid and that the space is not occupied by a piece. If the game_piece at the space has the default label, the space should be considered not occupied. If the space is both valid and not already occupied, then the space should be replaced by the parameter “piece” and the method should return 1. Otherwise, return 0.
int game_board_move_piece(struct game_board* game_board, int src_row, int src_col, int dest_row, int dest_col)	This method should validate that both the src and dest spaces are valid and that the dest space is not occupied. If both conditions pass, then the piece located at (src_row, src_col) should be moved to (dest_row, dest_col). The space at (src_row, src_col) should be replaced by the default game_piece. If this method moves the piece, return 1, otherwise return 0.

void game_board_print(struct game_board* game_board)	It prints information of the "board". It should show the list of pieces placed on the board using the game_piece_to_string function (it shows the first 3 characters of each piece). Use the following format:
	
	`The GameBoard`
	`-------------------`
	`--- Kin ---`
	`Paw --- ---`
	`--- --- Paw`
	
	Please see the sample output listed below.

After compiling the **homework_part_2.c** file, you need to execute it.

Sample Output: (the inputs entered by a user are shown in red)

```
Please enter the number of rows.
3
Please enter the number of columns.
3
Please enter a label for a new piece. Enter "Q" when done.
King
Please enter a row for the piece.
0
Please enter a column for the piece.
1
New piece "King" added.
Please enter a label for a new piece. Enter "Q" when done.Pawn
Please enter a row for the piece.
1
Please enter a column for the piece.
0
New piece "Pawn" added.
Please enter a label for a new piece. Enter "Q" when done.Pawn
Please enter a row for the piece.
2
Please enter a column for the piece.
2
New piece "Pawn" added.
Please enter a label for a new piece. Enter "Q" when done.Queen
Please enter a row for the piece.
4
Please enter a column for the piece.
5
Invalid row and/or column.
Please enter a label for a new piece. Enter "Q" when done.King
Please enter a row for the piece.
-1
Please enter a column for the piece.
2
Invalid row and/or column.
Please enter a label for a new piece. Enter "Q" when done.Q
The GameBoard
--------------------
--- Kin ---
Paw --- ---
--- --- Paw
Would you like to move a piece? Enter "Y" to move a piece.
Y
Please enter the piece's row.0
Please enter the piece's column.1
Please enter the piece's new row.0
Please enter the piece's new column.2
Piece moved to new space.
The GameBoard
--------------------
--- --- Kin
Paw --- ---
--- --- Paw
Would you like to move a piece? Enter "Y" to move a piece.
```

Grading Criteria for the part 2

05 pts: Every file contains header information
05 pts: adequate comment to explain every function
05 pts: consistent indentation and spacing
05 pts: it compiles
05 pts: The function game_piece_init_default is correct
05 pts: The function game_piece_init is correct
05 pts: The function game_piece_get_label is correct
05 pts: The function game_piece_to_string is correct
05 pts: The function game_board_init is correct
05 pts: The function game_board_is_space_valid is correct
05 pts: The function game_board_add_piece is correct
05 pts: The function game_board_move_piece is correct
05 pts: The function game_board_print is correct