CSE 240
Homework 2 – Programming with C
Due: Wednesday, September 6, 11:59 PM

1. What This Assignment Is About:

- Structures
- Functions
- Arrays of Primitive Values
- Arrays of Structs
- Recursion
- for and if Statements
- Selection 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) Create a file **homework_part_1.c**

b) Create a 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 1s in the even positions. **Hint: review pointers as parameters.**

c) Create a 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. **Hint: review pointers as parameters.**

d) Create a function **selection_sort** that receives as parameters an array of integers and the array size, and order the array element in descending order. Implement Selection Sort algorithm. It should be Selection Sort, not Bubble Sort, not Quick Sort, etc. If you do not remember selection sort, this link could be useful: https://goo.gl/hrAdMo

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

f) Copy the following main function in your class,

```
int main() {
    int a [10] = {3, 5, 6, 8, 12, 13, 16, 17, 18, 20};
    int b [6]= {18, 16, 19, 3 ,14, 6};
    int c [5]= {5, 2, 4, 3, 1};

    // testing initialize_array
    print_array(a, 10); // print: 3, 5, 6, 8, 12, 13, 16, 17, 18, 20
    initialize_array(a, 10);
    print_array(a, 10); // print: 1, 0, 1, 0, 1, 0, 1, 0, 1, 0

    // testing initialize_array
    print_array(b, 6); // print: 18, 16, 19, 3 ,14, 6
    selection_sort (b, 6);
    print_array(b, 6); // print: 19, 18, 16, 14, 6, 3

    // testing factorial
    printf("Factorail of 5 - %d\n", factorial (5)); //print: 120

    c[0] = factorial (c[0]);
    c[1] = factorial (c[2]);
    print_array(c, 5); // print: 120, 24, 4, 3, 1


    return 0;
}
```

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: selectionSort
08 pts: printArray
08 pts: initializeArray
08 pts: factorial

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

In this assignment, we will be making a program that reads in guests' information, and create a graduation ceremony seating with a number of rows and columns specified by a user. Then it will attempt to assign each guest to a seat in an auditorium.

Use the file **homework_part_2.c** (attached at the end of this document). Complete the file and include all the following requested code in the file homework_part_2.c

Step 1.

First, you need to create a structure **`guest`.** It should contain two variables, last_name (char [30]) and first_name (char [30]). In addition, the following functions should be defined.

Function	**Description**
void guest_init_default (struct guest *g)	Assign the default string " ??? " to both variables, last_name and first_name.
void guest_init (struct guest *g, char *info)	Use the **strtok function** to extract first name and last name from the variable guest, then assign them to each instance variable of the **guest** structure. An example of the input string is: David/Johnson
void guest_to_string (struct guest *g)	It prints the initial character of the first name, a period, the initial character of the last name, and a period. An example of such string for the guest David Johnson is: *D.J.*

Step 2.

You will be creating a structure called **`auditorium_seating`** in the same code file. The structure auditorium_seating will contain a 2-dimensional array called "seating" of **guest** type. Define the following functions:

Function	**Description**
void auditorium_seating_init (int rowNum, int columnNum, struct auditorium_seating *a)	It instantiates a two-dimensional array of the size "rowNum" by "columnNum" specified by the parameters inside the struct **a**. Then it initializes each guest element of this array using the **guest_init** function. So, each guest will have default values for its instance variables.

int assign_guest_at (int row, int col, struct auditorium_seating *a, struct guest* g)	The function attempts to assign the "g" to the seat at "row" and "col" (specified by the parameters of this function). If the seat has a default guest, i.e., a guest with the last name "???" and the first name "???", then we can assign the new guest "g" to that seat and the method returns true. Otherwise, this seat is considered to be taken by someone else, the method does not assign the guest and return **0** (false).
int check_boundaries (int row, int col, struct auditorium_seating *a)	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).
void auditorium_seating_to_string (struct auditorium_seating *a)	It prints information of the "seating". It should show the list of guests assigned to the seating using the **guest_to_string** function (it shows initials of each guest) and the following format: The current seating -------------------- D.J. ?.?. E.T.. ?.?. ?.?. S.W. T.C. A.T. ?.?. 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 bold)

Make sure that your program works at least with this scenario.

```
Please enter a number of rows for an auditorium seating.
3
Please enter a number of columns for an auditorium seating.
3
Please enter a guest information or enter "Q" to quit.
Mickey/Mouse
A guest information is read.
Mickey/Mouse
Please enter a row number where the guest wants to sit.
1
Please enter a column number where the guest wants to sit.
2
The seat at row 1 and column 2 is assigned to the guest M.M.

The current seating
--------------------
?.?. ?.?. ?.?.
?.?. ?.?. M.M.
?.?. ?.?. ?.?.

Please enter a guest information or enter "Q" to quit.
Daisy/Duck

A guest information is read.
Daisy/Duck
Please enter a row number where the guest wants to sit.
2
Please enter a column number where the guest wants to sit.
0

The seat at row 2 and column 0 is assigned to the guest D.D.

The current seating
--------------------
?.?. ?.?. ?.?.
?.?. ?.?. M.M.
D.D. ?.?. ?.?.

Please enter a guest information or enter "Q" to quit.
Clarabelle/Cow

A guest information is read.
Clarabelle/Cow

Please enter a row number where the guest wants to sit.
2
Please enter a column number where the guest wants to sit.
1

The seat at row 2 and column 1 is assigned to the guest C.C.

The current seating
--------------------
?.?. ?.?. ?.?.
?.?. ?.?. M.M.
D.D. C.C. ?.?.

Please enter a guest information or enter "Q" to quit.
Max/Goof
```

```
A guest information is read.
Max/Goof
Please enter a row number where the guest wants to sit.
0
Please enter a column number where the guest wants to sit.
0

The seat at row 0 and column 0 is assigned to the guest M.G.

The current seating
--------------------
M.G. ?.?. ?.?.
?.?. ?.?. M.M.
D.D. C.C. ?.?.

Please enter a guest information or enter "Q" to quit.
Horace/Horsecollar
A guest information is read.
Horace/Horsecollar

Please enter a row number where the guest wants to sit.
5
Please enter a column number where the guest wants to sit.
1
row or column number is not valid.
A guest Horace Horsecollar is not assigned a seat.
Please enter a guest information or enter "Q" to quit.
Sylvester/Shyster

A guest information is read.
Sylvester/Shyster
Please enter a row number where the guest wants to sit.
2
Please enter a column number where the guest wants to sit.
0
The seat at row 2 and column 0 is taken.
Please enter a guest information or enter "Q" to quit.
Snow/White

A guest information is read.
Snow/White
Please enter a row number where the guest wants to sit.
-1

Please enter a column number where the guest wants to sit.
0

row or column number is not valid.
A guest Snow White is not assigned a seat.
Please enter a guest information or enter "Q" to quit.
Jiminy/Criket

A guest information is read.
Jiminy/Criket
Please enter a row number where the guest wants to sit.
0
Please enter a column number where the guest wants to sit.
2

The seat at row 0 and column 2 is assigned to the guest J.C.

The current seating
--------------------
M.G. ?.?. J.C.
?.?. ?.?. M.M.
```

```
D.D. C.C. ?.?.

Please enter a guest information or enter "Q" to quit.
Q
```

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
10 pts: The functions guest_init are correct
05 pts: The function guest_to_string method is correct
10 pts: The function auditorium_seating_init is correct
10 pts: The function assign_guest_at (int, int, g) method is correct
05 pts: The function check_boundaries(int, int) method is correct
05 pts: The function auditorium_seating_to_string method is correct

```c
#include <stdio.h>
struct guest {
 char last_name[30] ;
 char first_name[30];
};
struct auditorium_seating {
 struct guest **seating;
};
void guest_init_default (struct guest *g ) {}
void guest_init (struct guest *g, char *info) {}
void guest_to_string (struct guest *g ) {}
void auditorium_seating_init (int rowNum, int columnNum, struct auditorium_seating *a ) {}
int assign_guest_at (int row, int col, struct auditorium_seating *a, struct guest* g) {}
int check_boundaries (int row, int col, struct auditorium_seating *a) {}
void auditorium_seating_to_string (struct auditorium_seating *a ) {}

void main() {
  struct auditorium_seating auditorium_seating;
  struct guest temp_guest;

  int row, col, rowNum, columnNum;

  char guest_info[30];

  // Ask a user to enter a number of rows for an auditorium seating
  printf ("Please enter a number of rows for an auditorium seating.");
  scanf ("%d", &rowNum);

  // Ask a user to enter a number of columns for an auditorium seating
  printf ("Please enter a number of columns for an auditorium seating.");
  scanf ("%d", &columnNum);

  // auditorium_seating
  auditorium_seating_init(rowNum, columnNum, &auditorium_seating);

  printf("Please enter a guest information or enter \"Q\" to quit.");

  /*** reading a guest's information ***/
  scanf ("%s", guest_info);

  /* we will read line by line **/
  while (1 /* change this condition*/ ){
    printf ("\nA guest information is read.");
    // printing information.
    printf ("%s", guest_info);
    // guest
    guest_init (&temp_guest, guest_info);
    // Ask a user to decide where to seat a guest by asking
    // for row and column of a seat

    printf ("Please enter a row number where the guest wants to sit.");
    scanf("%d", &row);

    printf("Please enter a column number where the guest wants to sit.");
    scanf("%d", &col);
        // Checking if the row number and column number are valid
        // (exist in the theatre that we created.)
    if (check_boundaries(row, col, &auditorium_seating) == 0) {
      printf("\nrow or column number is not valid.");
      printf("A guest %s %s is not assigned a seat.", temp_guest.first_name, temp_guest.last_name);
    } else {
      // Assigning a seat for a guest
      if (assign_guest_at(row, col, &auditorium_seating, &temp_guest) == 1){
        printf("\nThe seat at row %d and column %d is assigned to the guest",row, col);
        guest_to_string(&temp_guest);
        auditorium_seating_to_string(&auditorium_seating);
      } else {
        printf("\nThe seat at row %d and column %d is taken.", row, col);
      }
    }
    // Read the next guestInfo
    printf ("Please enter a guest information or enter \"Q\" to quit.");
    /*** reading a guest's information ***/
    scanf("%s", guest_info);
  }

}
```