C Programming Language
#Project - 10

This project consist of four sub-parts. For each part, you need to make performance comparisons in terms of execution time and print them with the following format (Execution times should be represented with milliseconds). Remember that, if you are asked to implement your solution by using a data structure, then, you can only use the data structures which you have already learned in the class. Please note that, you are not allowed to use dynamic arrays in your data structure implementations. The main function is also given at the end of the document.

```
The Performance result reports should be seem as following:

Structures     Stack      Queue      BST

Exec. Time      1.24       0.89       0.06
```

Part 1. Write a function that takes three data structure as parameters and a data array to fill them with the values of the data array. The data structures should be queue, stack and binary search tree. Tract the execution time of filling time for each data structure and report the total execution time of filling the structures separately. The function prototype is:

```
void fill_structures(stack ** stack_, queue ** queue_, bst ** bst_, int
                                data[20])
```

Part 2. Write a function that takes the data structures parameters which were filled on part one as and print them in descending order. Report the total of structure ordering and printing time for each separately. The function prototype is:

```
void ordered_print(stack * stack_, queue * queue_, bst * bst_)
```

Part 3. Write a function that takes the data structures which were filled on part one as parameters and also an integer parameter to search in the structures. The function prints all the founded items and also number of steps to find the result, the function should report if there is no result. In example; let the queue be (tail)1→3→5→2→11→18](head) and we want to search 3, then the result of the search should be; "1 result founded on 5. step." for queue structure and so on. Report the total of search time for each structure separately. The function prototype is:

```
void search(stack * stack_, queue * queue_, bst * bst_, int val_to_search)
```

Part 4. Write a function that takes the data structures which were filled on part one as parameters and print them in a specific traversal method. The traversal method follows the basic rule:

Start with the max value of the structure and go on with the min value and after that, again the max value and the min value, so on and so forth.

In example; let the stack be 3, 5, 6, 2, 1, 7 (top), then the function should print ; 7, 1, 6, 2, 5, 3. Report the total of traverse time for each structure separately. The function prototype is:

```
void special_traverse(stack * stack_, queue * queue_, bst * bst_)
```

The main function:

```
int main()

{

    int data[20]={5, 2, 7, 8, 11, 3, 21, 7, 6, 15, 19, 35, 24, 1, 8, 12, 17,
8, 23, 4};

    bst * bst_;

    queue * queue_;

    stack * stack_;

    fill_structures(&stack_, &queue_, &bst_, data);

    ordered_print(stack_, queue_, bst_);

    search(stack_, queue_, bst_, 5);

    special_traverse(stack_, queue_, bst_);

    return 0;

}
```