Problem 4

I. Answer the following questions on logic circuits. In circuit diagrams, you may use AND and OR which have three or more inputs. To fix an input value to 0 or 1, put “0” or “1” at the input.

(1) Using the symbols in Fig. 1, draw a circuit of a half adder, which outputs sum $S$ and carry $C_{\text{out}}$ of 1-bit inputs $A$ and $B$.

A half adder cannot receive a carry from the lower digit. An adder that can add a carry from the lower digit is called a full adder.

(2) Give the truth table of a full adder that has an input $C_{\text{in}}$ for the carry from the lower digit, in addition to the inputs and outputs of the half adder in Question (1).

(3) Design a circuit that gives $C_{\text{out}}$ of the full adder in Question (2) using a Karnaugh map, and draw the circuit using the symbols in Fig. 1.

(4) The symbol in Fig. 2 represents a full adder. Consider a circuit with a 1-bit input signal $I$ and a 2-bit output signal $O_1O_0$. The output $O_1O_0$ changes in the order as shown below every time when the input $I$ changes from 0 to 1.

\[
\uparrow \quad 00 \rightarrow 01 \rightarrow 10 \rightarrow 11
\]

Draw the circuit by using the symbols in Fig. 1 and Fig. 2. The clock inputs $\text{CLK}$ of the all D-FF must be connected to the input $I$.
II. Answer the following questions on the binary search tree. The binary search tree is a binary tree which has the following properties; a value is stored in each node in the tree, the value at each node is always greater than any values stored in that node’s left sub-tree, and always less than any values in that node’s right sub-tree. Here, the values in the tree are integers, and there is no duplication among them.

(1) Consider the procedure to search a binary search tree for a specific value. When \( n \) denotes the number of nodes in the binary search tree, what is the order of the average time complexity of the searching algorithm? In addition, describe the reason briefly.

(2) Consider completing Program 1 for the search algorithm. Fill in the blank (A) using C language. Note that function ‘search’ in Program 1 takes the pointer ‘r’ to the root of the tree and the key ‘key’ for searching as input. It returns the pointer to the node which is found when the search succeeds, or NULL when the search fails.

(3) Rewrite function ‘search’ in Program 1 as a recursive algorithm.

(4) By modifying the program from either Question (2) or Question (3), write function ‘insert’ using C language, where a new value ‘v’ is added to the binary search tree whose root node is ‘r’.

/* Program 1 */

typedef struct node{
    int value;
    struct node *left, *right;
} Node;

Node* search(Node* r, int key){
    Node* current_node = r;
    while(current_node != NULL){
        int diff = key - current_node->value;

    }
    return current_node;
}