Problem 4

I. Answer the following questions on logic circuits. Fig. 1 shows the symbols of logic gates used in circuit schematics. Assume that the propagation delay is 50 ps for the AND gate and for the OR gate, 30 ps for the NOT gate and 80 ps for the XOR gate, and that the wiring delay is negligible. A critical path (the path with the longest propagation delay between an input and an output) should be shown together with the pair of input and output ports. If there are multiple critical paths, all of them should be shown.

(1) Fig. 2 illustrates a full adder circuit which adds two inputs A, B together with a carry input CI to generate a sum S and a carry output CO.

(1-i) X in Fig. 2 is a half adder which generates D (the sum of A and B) and a carry output E. Write down the truth table for D and E, and draw the circuit schematic of X using only AND, OR and NOT gates shown in Fig. 1. Find the critical path and its propagation delay.

(1-ii) Draw a circuit schematic of X which has a smaller critical path propagation delay when all the gates in Fig. 1 can be used. Then, find the critical path and its propagation delay.

(1-iii) Write down the truth table for Y and Z, with CI, D and E as the inputs, and S and CO as the outputs. Then draw a circuit schematic for Y and Z whose propagation delay on the critical path is minimal. All the gates in Fig. 1 can be used. Find the critical path and its propagation delay of the full adder when the circuit designed for Question (1-ii) is used for X.

Hereafter, the circuits designed for Questions (1-ii) and (1-iii) are used as a half adder and a full adder, respectively.

(2) Using full adders, design an unsigned 3-bit adder which has two 3-bit inputs $a_2\,a_1\,a_0$ and $b_2\,b_1\,b_0$, a carry input ci, a 3-bit sum $s_2\,s_1\,s_0$, and a carry output co. Illustrate its schematic and find the critical path and its propagation delay. Use the symbol in Fig. 3 for a full adder.

(3) Consider an unsigned multiplier shown in Fig. 4, which has a 3-bit input $a_2\,a_1\,a_0$ and a 2-bit input $b_1\,b_0$.

(3-i) State the bit width N of the product p.

(3-ii) Design and draw the circuit for the unsigned multiplier as shown in Fig. 4 by using at least one half adder and one full adder. The propagation delay of its critical path must be kept as small as possible. All the symbols in Fig. 1 and Fig. 3 can be used. Find the critical path and its propagation delay.

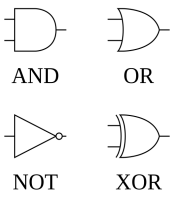

Fig. 1

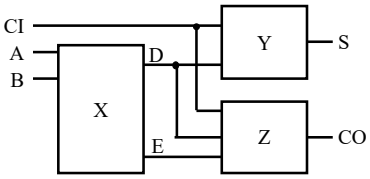

Fig. 2

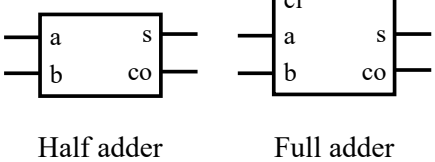

Fig. 3

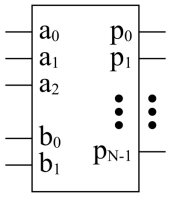

Fig. 4

II. Consider a C language program that uses a binary search tree for managing the student identification number (sid) of students. A binary search tree (Fig. 5) is a node-based tree data structure in which each node has a maximum of two (left and right) child nodes. The value of a node is always larger than that of its left child node, and smaller than that of its right child node. A node managing an sid is represented by the structure in Program 1. Answer the following questions.

(1) Function `add()` in Program 2 adds a node with a new sid. Complete the function by filling in the blanks A, B and C.

(2) Consider adding sids to an empty tree using `add()` in the following orders. Illustrate the resultant binary search trees following the example shown in Fig. 5.

(2-i) sid = 1040, 1042, 2001, 2004, 2010, 2012

(2-ii) sid = 2001, 1042, 2010, 1040, 2004, 2012

(3) The function `enumerate()` in Program 3 enumerates sids in an ascending order. Complete the function by filling in the blanks D, E, F and G.

(4) Briefly explain the procedures to remove a node from a binary search tree for each of the following three cases: the number of its child nodes is zero, one and two.

(5) Briefly explain the behavior and the problem of the function `add()` when adding a node with an sid that already exists in the binary search tree.

(6) Briefly explain the advantages and disadvantages of the binary search tree, compared to the scheme in which a new sid is stored in a pre-allocated and sufficiently large array from the head in the order of the addition. Explain from the viewpoint of the order of the time complexity for each of the searching and adding process.


```
/* Program 1 */
typedef struct node {
    int sid;
    struct node *left;
    struct node *right;
} node_t;
```

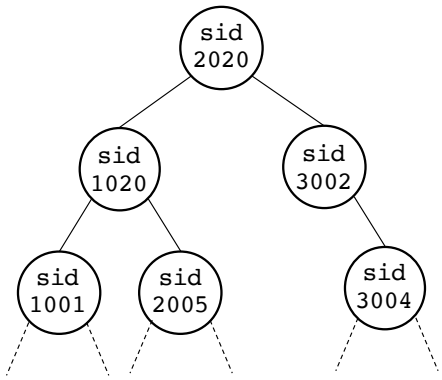

Fig. 5

```
/* Program 2 */
node_t *add(node_t *root, node_t *new){
    if(root == NULL){
        return [      A      ];
    }

    if(root->sid > new->sid){
        [      B      ];
    }
    else if(root->sid < new->sid){
        [      C      ];
    }
    return root;
}
```

```
/* Program 3 */
void enumerate(node_t *root){
    if( [      D      ] ){
        [      E      ];
    }
    printf("%d¥n", root->sid);
    if( [      F      ] ){
        [      G      ];
    }
    return;
}
```