問題４

I. 論理回路に関する以下の問に答えよ．回路図に使用する論理ゲートを図 1 に示す．AND, OR ゲートの伝搬遅延時間がそれぞれ 50 ps, NOT ゲートの伝搬遅延時間が 30 ps, XOR ゲートの伝搬遅延時間が 80 ps で，配線遅延は無視できるものとする．なお，クリティカルパス（入力から出力までの伝搬遅延時間が最も大きい経路）は入力と出力の組で示し，複数ある場合はすべて示すこと．

(1) 図 2 は，入力 A, B を，キャリー（桁上げ）入力 CI を考慮して加算し，和 S とキャリー出力 CO を生成する全加算器である．

(1-i) 図 2 の X は，A, B の和 D とキャリー出力 E を生成する半加算器である．D, E の真理値表を示せ．また，図 1 の AND, OR, NOT ゲートのみを用いて X の回路を示し，クリティカルパスとその遅延を示せ．

(1-ii) 図 1 のすべてのゲートが使用できるとき，クリティカルパスの伝搬遅延時間がより小さい X の回路を示せ．また，クリティカルパスとその伝搬遅延時間を示せ．

(1-iii) Y，Z について，CI, D, E を入力，S, CO を出力とする真理値表を示し，クリティカルパスの伝搬遅延時間が最小となる回路を示せ．図 1 のすべてのゲートを使用してよい．また，X に問(1-ii)の回路を使用した場合のこの全加算器のクリティカルパスとその伝搬遅延時間を示せ．

以下では，半加算器は問(1-ii)で作成したもの，全加算器は問(1-iii)で作成したものを使用する．

(2) 全加算器を用いてキャリー入力を持つ符号なし 3 ビット加算器を作成し，回路図を示せ．また，クリティカルパスとその伝搬遅延時間を示せ．2 組の 3 ビットの入力を $a_2\,a_1\,a_0$ と $b_2\,b_1\,b_0$，キャリー入力を ci，和を $s_2\,s_1\,s_0$，キャリー出力を co とする．全加算器の記号は図 3 のものを用いよ．

(3) 図 4 に示す，3 ビットの入力 $a_2\,a_1\,a_0$ と 2 ビットの入力 $b_1\,b_0$ とを持つ符号なし乗算器を考える．

(3-i) 積 p のビット幅 N を答えよ．

(3-ii) 半加算器，全加算器を少なくともひとつずつ使用してこの乗算器を作成し，回路図を示せ．ただし，クリティカルパスの伝搬遅延時間をなるべく小さくすること．図 1，図 3 のすべての記号を用いてよい．また，クリティカルパスとその伝搬遅延時間を示せ．

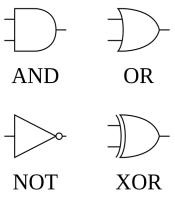

図 1

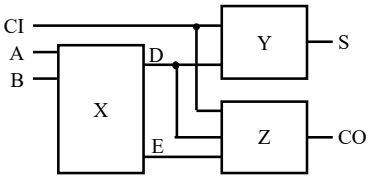

図 2

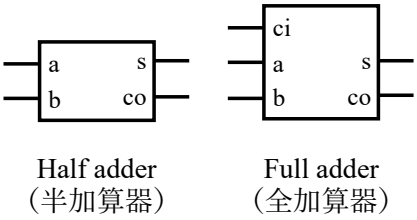

図 3

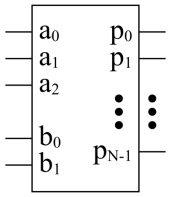

図 4

Sample

II. 学生の学籍番号（sid）を，2 分探索木を用いて管理する C 言語のプログラムを考える．2 分探索木は，図 5 のように，各ノードに最大 2 個の子ノードが接続された木構造であり，ノードの値は，左の子ノードの値より大きく右の子ノードの値より小さいという制約がある．学籍番号を持つノードをプログラム 1 の構造体で表現する．以下の問に答えよ．

(1) プログラム 2 に，新しい sid を持つノードを追加する関数 `add()` を示す．空欄 A, B, C を埋めて完成させよ．

(2) 空の木に `add()` を使い以下の順で新しいノードを追加したときの 2 分探索木を図 5 の例に倣って書け．

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

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

(3) プログラム 3 に sid を小さい順に列挙する関数 `enumerate()` を示す．空欄 D, E, F, G を埋めて完成させよ．

(4) ノードを 2 分探索木から削除する手順を，削除したいノードの子ノードが 0 個，1 個，2 個の場合について簡潔に説明せよ．

(5) 既に 2 分探索木上に存在するノードと同じ sid を持つノードを追加しようとした時の `add()` の振る舞いと問題点を簡潔に説明せよ．

(6) 追加する sid をあらかじめ用意された十分に大きな配列に先頭から追加順に格納する方法と比べ，2 分探索木を用いる利点と欠点をそれぞれ簡潔に述べよ．その際，探索，要素の追加のそれぞれに必要な計算量のオーダーについて言及せよ．

Sample

```
/* プログラム1 */
typedef struct node {
    int sid;
    struct node *left;
    struct node *right;
} node_t;
```

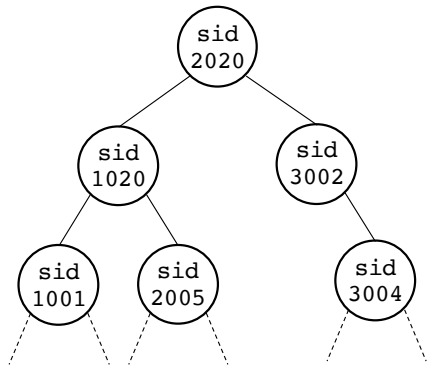

図5

```
/* プログラム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;
}
```

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