1 Purpose

Using high level program language like C or C^{++} to implement lab 1 to lab 4.

2 Principles

2.1 Lab 1

The key to this experiment is to determine parity, and I used the "and operator". Let the parameter and "1" for the bit "and" operation, if the result is not 0, it indicates that the parameter is odd, and then perform the corresponding operation.

2.2 Lab 2

The key of this experiment lies in the realization of multiplication operation and modulo operation. I realized multiplication operation by loop the basic addition operation and modulo operation by loop the basic subtraction operation.

Listing 1: Multiplication

```
int16_t multi(int16_t a , int16_t b){
    int16_t result = a;
    while ((b-- - 1) > 0){
        result += a;    // Lopp b times result += a
    }
    return result;
}
```

Listing 2: Module

```
int16_t module(int16_t A , int16_t B, int16_t *reminder){
    int16_t result = 0;
    while ((A -= B) >= 0){
        result++;    //Excute A - B untill A < B
    }
    *reminder = A + B;    //Return A%B
    return result;    //Return A/B
}
```

2.3 Lab 3

There's nothing complicated about this experiment, just be careful to loop through all the strings. In addition, different cases should be handled separately, such as the current character is the same or different, one of the string traversal ends, and so on.

2.4 Lab 4

The key of this experiment lies in the design of `Remove` function and `Put` function. According to the requirements, there are two mathematical expressions as follows.

$$Remove(n) = \begin{cases} \text{Do nothing} & n = 0 \\ \text{Remove the } 1^{th} \text{ ring} & n = 1 \\ Remove(n-2) + \text{Remove the } n^{th} \text{ ring off} + Put(i-2) + Remove(i-1) & n > 1 \end{cases}$$

$$Put(n) = \begin{cases} \text{Do nothing} & n = 0 \\ \text{Put the } 1^{th} \text{ ring on} & n = 1 \\ Put(n-1) + R(n-2) + \text{Put the } n^{th} \text{ ring on} + Put(n-2) & n > 2 \end{cases}$$

Once I have defined the two functions above, I can implement them in a high-level program language as follows.

```
//Remove function
void Remove(int16_t *state, int16_t *move, int16_t *memory, int16_t n){
    int16_t mask = 1;
    if (n == 0) return ;    //Do nothing
    if (n == 1){            //Remove the first ring off
        *state += 1;
        memory[*move] = *state;
        *move += 1;
        return ;
    }else{
        Remove (state, move, memory, n - 2);    //Remove(n-2)
        for (int i = 1 ; i < n ; i++){          //Using mask
            mask += mask;
        }
        *state += mask;                         //Remove the nth ring off
        memory[*move] = *state;                 //Store the new state
        *move += 1;                             //Recording the moves
        Put(state, move, memory, n - 2);        //Put(n-2)
        Remove(state, move, memory, n - 1);     //Remove(n-1)
    }
}

//Put function
void Put(int16_t *state, int16_t *move, int16_t *memory,int16_t n){
    int16_t mask = 1;
    if (n == 0) return ;    //Do nothing
    if (n == 1){            //Put the first ring on
        *state -= 1;
        memory[*move] = *state;
        *move += 1;
        return ;
    }else{
        Put (state, move, memory, n - 1);       //Put(n-1)
        Remove (state, move, memory, n - 2);    //Remove(n-2)
        for (int i = 1 ; i < n ; i++){          //Using mask
            mask += mask;
        }
        *state -= mask;                         //Put the nth ring on
        memory[*move] = *state;                 //Store the new state
        *move += 1;                             //Recording the moves
        Put(state, move, memory, n - 2);        //Put(n-2)
    }
}
```

3 Procedure

3.1 Lab 1

```
int16_t lab1(int16_t n) {
    int16_t mask = 1;                       //Mask
    int16_t result = 0;                     //Number of 0s
    if ((mask & n) > 0){                    //Judge if n is odd or even
        for (int i = 0 ; i < 16 ; i++){
            if ((mask & n) == 0){           //Counting the 0s
                result++;
            }
            mask += mask;
        }
    }else{
        for (int i = 0; i < 16; i++){
            if ((mask & (0 - n)) == 0){     //Counting the 0s
                result++;
            }
            mask += mask;
        }
    }

    return result + STUDENT_ID_LAST_DIGIT;  //Return the result
}
```

3.2 Lab 2

```
int16_t lab2(int16_t n) {
    int16_t dn = 0;         //Recording the operator
    int16_t vn = 3;         //Recording vn

    int16_t index = 1;
    int16_t reminder;
    while ((index++ - n) != 0){
        if (dn == 0){
            module((multi(vn, 2) + 2), 4096, &vn);//vn = (vn*2 + 2)%4096
        }else{
            module((multi(vn, 2) - 2), 4096, &vn);//vn = (vn*2 - 2)%4096
        }
        module(vn, 8, &reminder);
        if (reminder == 0){         //If vn can divided by 8
            dn = 1 - dn;            //Change dn
            continue;
        }
        module(vn, 10, &reminder);
        if ((reminder - 8) == 0){   //If the last digit of vn is 8
            dn = 1 - dn;            //Change dn
        }
    }

    return vn;                      //Return the result
}
```

3.3 Lab 3

```
int16_t lab3(char s1[], char s2[]) {
    int16_t difference = 0;          //The difference between characters
    int index = 0;
    while (1){
        difference = s1[index] - s2[index];     //Calculate difference
        if (difference != 0){        //If the characters are not the same
            break;
        }
        if (s1[index] == 0 || s2[index] == 0){  //If s1 or s2 are done
            break;
        }
        index++;
    }

    return difference;           //Return the difference
}
```

3.4 Lab 4

```
int16_t lab4(int16_t *memory, int16_t n) {
    int16_t state = 0;       //Inite the state as 0
    int16_t move = 0;        //Recording the numbers of moves
    Remove (&state, &move, memory, n);      //Call for Remove(n)
    return move;         //Return the numbers of moves
}
```

4 Results

The outputs are as expected.

```
===== lab1 =====
21                          Input: 5
===== lab2 =====
786                         Input: 9
===== lab3 =====
115                         Input: DsTAs DsTA
===== lab4 =====
0000000000000001            Input: 3
0000000000000101
0000000000000100
0000000000000110
0000000000000111
```