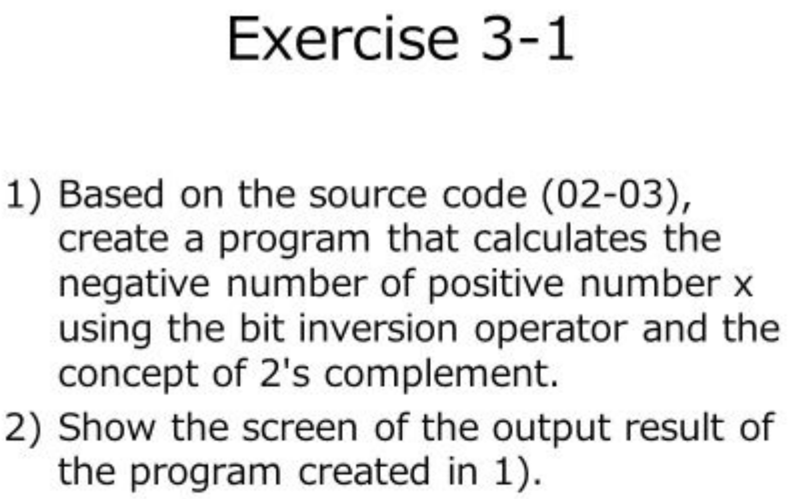
2020 College of Engineering Systems

Introduction to programming B 【exercise-week03】

Student ID（202018021） Name（Fu, Ziyu）

Answer the following questions. You can change the answer space freely.

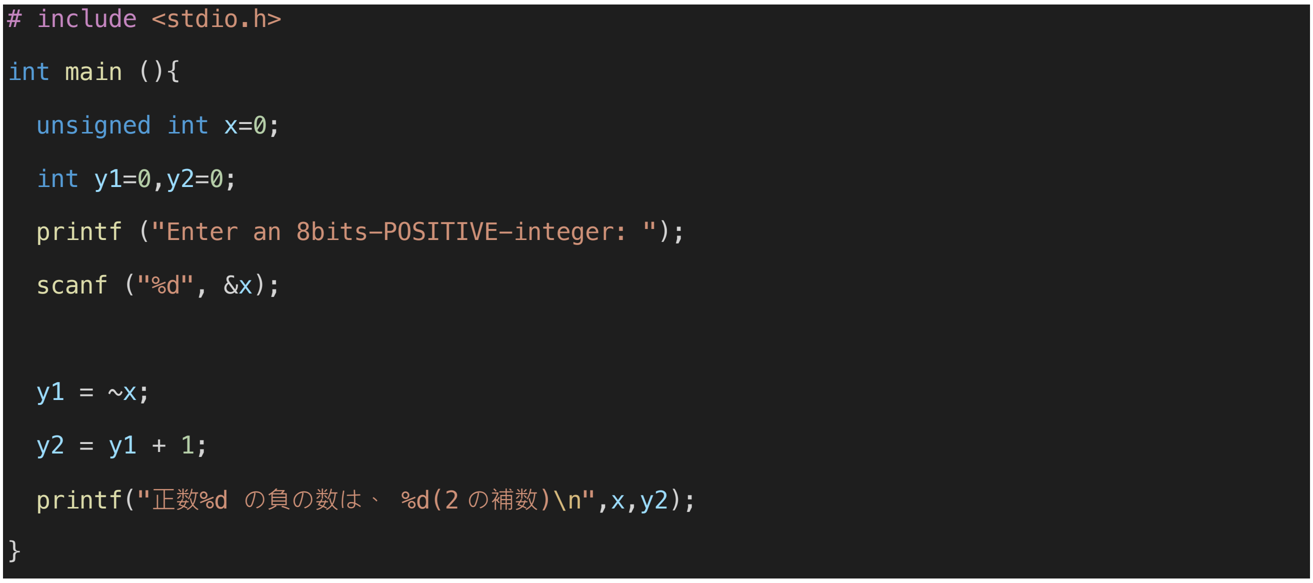
【Exercise 3-1】



Exercise 3-1

1) Based on the source code (02-03), create a program that calculates the negative number of positive number x using the bit inversion operator and the concept of 2's complement.
2) Show the screen of the output result of the program created in 1).

1）Program:

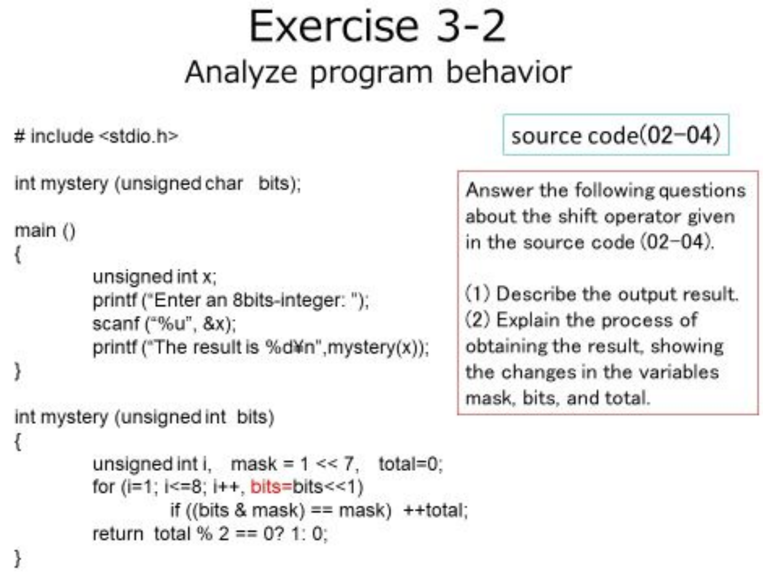


```
# include <stdio.h>

int main (){

  unsigned int x=0;

  int y1=0,y2=0;

  printf ("Enter an 8bits-POSITIVE-integer: ");

  scanf ("%d", &x);


  y1 = ~x;

  y2 = y1 + 1;

  printf("正数%d の負の数は、 %d(2 の補数)\n",x,y2);

}
```

2）Sample output:

```
Enter an 8bits-POSITIVE-integer: 89
正数 89 の負の数は、 -89(2 の補数)
```

【Exercise 3-2】



Exercise 3-2

Analyze program behavior

source code(02−04)

```
# include <stdio.h>

int mystery (unsigned char   bits);

main ()
{
            unsigned int x;
            printf ("Enter an 8bits-integer: ");
            scanf ("%u", &x);
            printf ("The result is %d¥n",mystery(x));
}

int mystery (unsigned int  bits)
{
            unsigned int i,    mask = 1 << 7,    total=0;
            for (i=1; i<=8; i++, bits=bits<<1)
                        if ((bits & mask) == mask)  ++total;
            return  total % 2 == 0? 1: 0;
}
```

Answer the following questions about the shift operator given in the source code (02−04).

(1) Describe the output result.
(2) Explain the process of obtaining the result, showing the changes in the variables mask, bits, and total.

1) The possible outputs are 1 or 0.

   Here are two cases where each output is obtained:

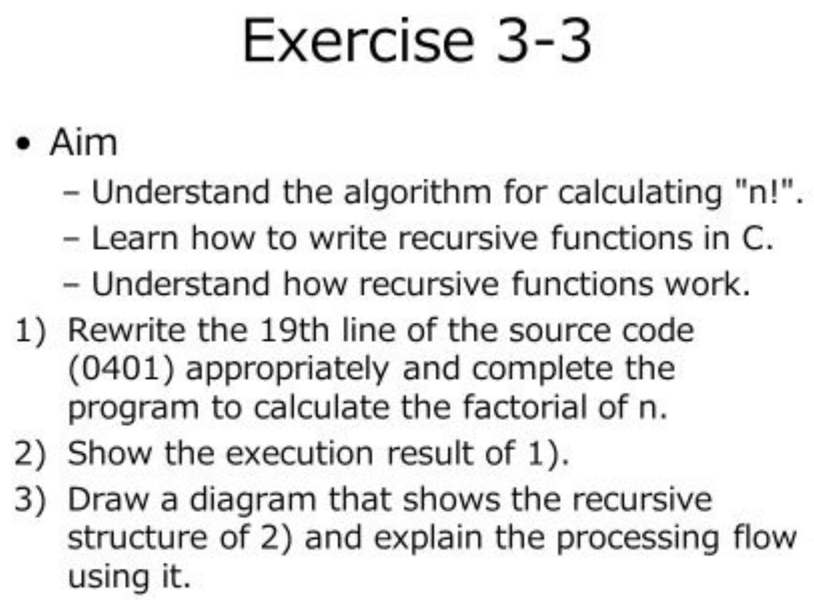
   ```
   Enter an 8bits-integer: 50
   The result is 0
   Enter an 8bits-integer: 51
   The result is 1
   ```

2) The `mask` variable is `1 left shift 7`, which is `1000 0000`, an 8-bit binary representation with only the leading bit as 1. The `for` loop in `mystery` runs 8 times, left shifting the bits variable by 1 seven times (once per loop after the first run). `total` counts the number of times where the 2^7 bit is 1 in each left-shift (by calculating bitwise AND between the left-shifted number and `1000 0000`). In other words, because the input has to be an 8-bit integer, the function actually counts the amount of 1's in the input number's binary form. The function returns 1 if total is divisible by 2, and 0 if it isn't.

   If input is 50, its binary form is `0011 0010`. the bitwise AND returns `1000 0000` at `50 << 2 = 1100 1000`, `50 << 3 = 1 1001 0000`, and `50 << 6 = 1100 1000` 0000. `total` is 3, non-divisible by 2, hence the function returns 0.

   If input is 51, its binary form is `0011 0011`. the bitwise AND returns `1000 0000` at `50 << 2 = 1100 1100`, `50 << 3 = 1 1001 1000`, `50 << 6 = 1100 1100 0000`, and `50 << 7 = 1 1001 1000 0000`. `total` is 4, divisible by 2, hence the function returns 1.

【Exercise 3-3】



Exercise 3-3

- Aim
  - Understand the algorithm for calculating "n!".
  - Learn how to write recursive functions in C.
  - Understand how recursive functions work.

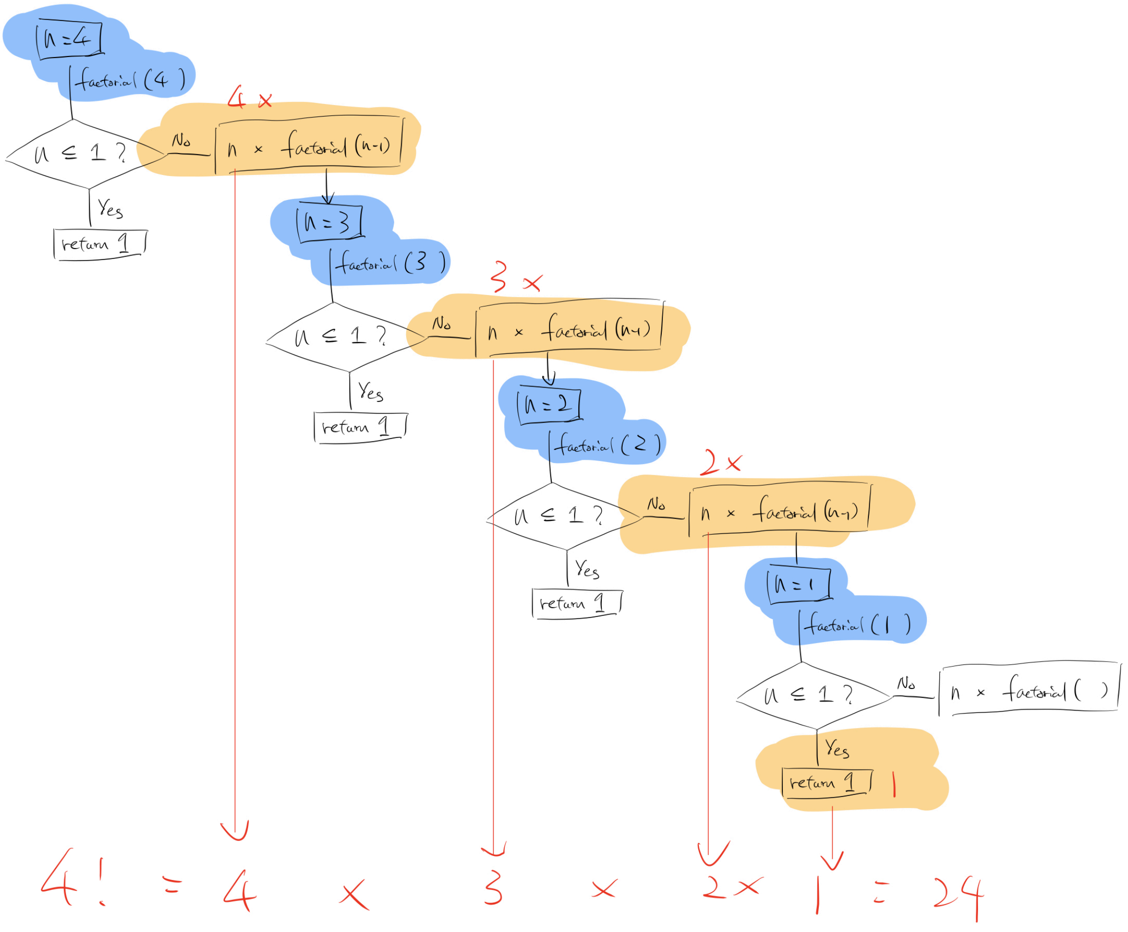
1) Rewrite the 19th line of the source code (0401) appropriately and complete the program to calculate the factorial of n.
2) Show the execution result of 1).
3) Draw a diagram that shows the recursive structure of 2) and explain the processing flow using it.

1) `????` should be `n-1`.

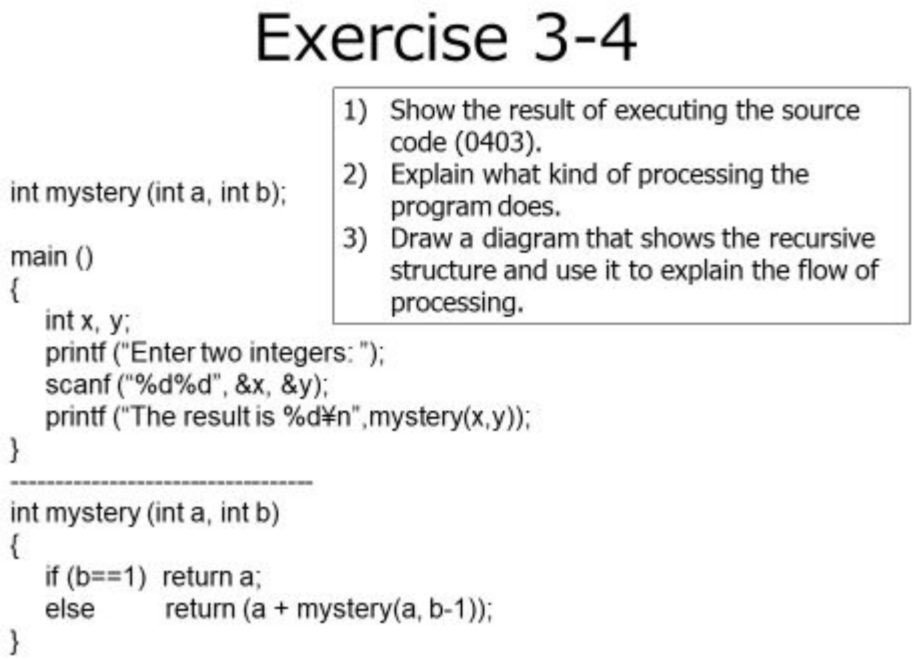
2) Output:

```
1! = 1
2! = 2
3! = 6
4! = 24
```

3)



【Exercise 3-4】



Exercise 3-4

1) Show the result of executing the source code (0403).
2) Explain what kind of processing the program does.
3) Draw a diagram that shows the recursive structure and use it to explain the flow of processing.

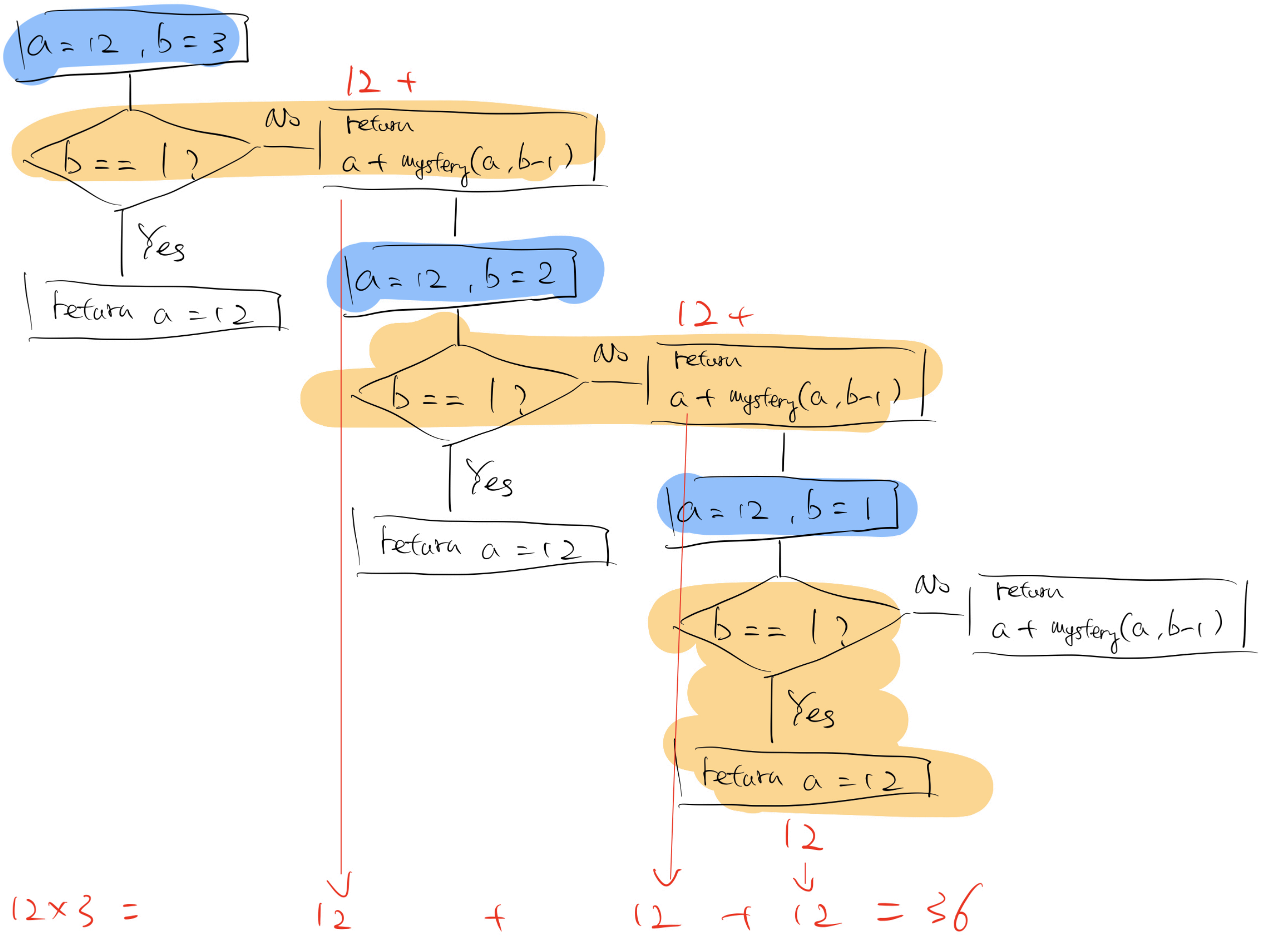
```
int mystery (int a, int b);

main ()
{
    int x, y;
    printf ("Enter two integers: ");
    scanf ("%d%d", &x, &y);
    printf ("The result is %d¥n",mystery(x,y));
}
------------------------------------
int mystery (int a, int b)
{
    if (b==1)  return a;
    else        return (a + mystery(a, b-1));
}
```

1) Sample output:

```
Enter two integers: 12 3
The result is 36
```

2) It calculates the product of a times b by adding up a, b times.

3)

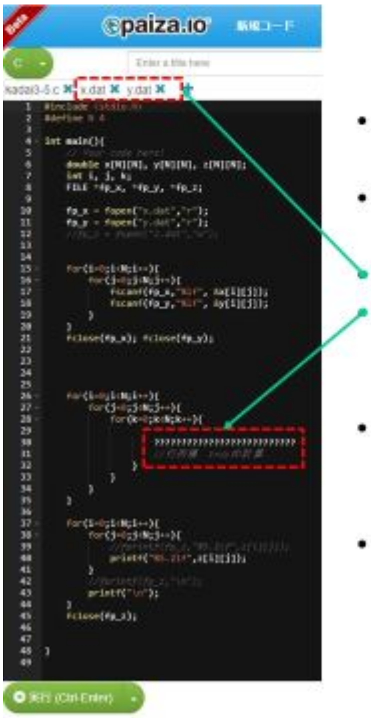


【Exercise 3-5】

Exercise 3-5

- The exercise is to create and execute the source code of the file I/O introduced in the lecture.

See the next page.



```
#include <stdio.h>
#define N 4

int main(){
    // Your code here!
    double x[N][N], y[N][N], z[N][N];
    int i, j, k;
    FILE *fp_x, *fp_y, *fp_z;

    fp_x = fopen("x.dat","r");
    fp_y = fopen("y.dat","r");
    //fp_z = fopen("z.dat","w");


    for(i=0;i<N;i++){
        for(j=0;j<N;j++){
            fscanf(fp_x,"%lf", &x[i][j]);
            fscanf(fp_y,"%lf", &y[i][j]);
        }
    }
    fclose(fp_x); fclose(fp_y);




    for(i=0;i<N;i++){
        for(j=0;j<N;j++){
            for(k=0;k<N;k++){

                ??????????????????????????
                //行列積 z=xyの計算
                }
            }
        }
    }

    for(i=0;i<N;i++){
        for(j=0;j<N;j++){
            //fprintf(fp_z,"%5.2lf",z[i][j]);
            printf("%5.2lf",z[i][j]);
        }
        //fprintf(fp_z,"\n");
        printf("\n");
    }
    fclose(fp_z);


}
```

Exercise 3-5

- It is kadai3-5.c. (It will not be uploaded.)
- Please copy while looking at the source code on the left.
- Use x.dat, y.dat (drag and drop)
- Please describe considering "??" (//Calculation of matrix product z=xy)
- Please write the source code in text on the answer sheet. (not capture)
- In paiza, w of fopen cannot be used, so after pressing the execute button, screen capture the output, or copy and paste it into the text.

```c
#include <stdio.h>
#define N 4
void main(){
    double x[N][N], y[N][N], z[N][N];
    int i, j, k;
    FILE *fp_x, *fp_y;

    fp_x = fopen("x.dat","r");
    fp_y = fopen("y.dat","r");

    for(i = 0; i < N; i++){
        for(j = 0; j < N; j++){
            fscanf(fp_x, "%lf",&x[i][j]);
            fscanf(fp_y, "%lf",&y[i][j]);
        }
    }
    fclose(fp_x);
    fclose(fp_y);

    for(i = 0; i < N; i++){
        for(j = 0; j < N; j++){
            for(k = 0; k < N; k++){
                z[i][j] += x[i][k] * y[k][j];
            }
        }
    }

    for(i = 0; i < N; i++){
        for(j = 0; j < N; j++){
            printf("%5.2lf  ", z[i][j]);
        }
        printf("\n");
    }
}
```

(See outputs on the next page.)

Output:

```
1.00   2.00   3.00   4.00
2.00   5.00   6.00   7.00
3.00   6.00   8.00   9.00
4.00   7.00   9.00  10.00
```

Notes:

Since we don't need to output the calculation results to a file, there is no point in printing from `fprintf`.