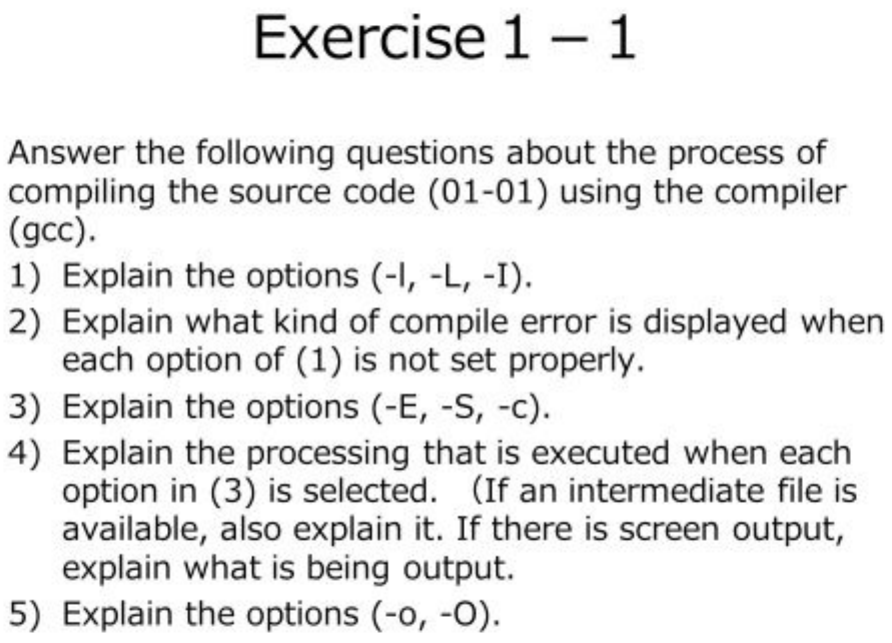
2020　College of Engineering Systems
Introduction to programming B　【exercise-week01】

Student ID（202018021）　Name（Fu, Ziyu）

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

【Exercise 1-1】



Exercise 1 – 1

Answer the following questions about the process of compiling the source code (01-01) using the compiler (gcc).

1) Explain the options (-l, -L, -I).
2) Explain what kind of compile error is displayed when each option of (1) is not set properly.
3) Explain the options (-E, -S, -c).
4) Explain the processing that is executed when each option in (3) is selected.　(If an intermediate file is available, also explain it. If there is screen output, explain what is being output.
5) Explain the options (-o, -O).

1) **`-I`** specifies a path to the included headers files.
   `-l` specifies the name of the library used when linking.
   `-L` specifies the directory to the library.
2) A linker error like “undefined reference” would appear. The linking process will not be successful.
3) **`-E`** preprocesses the file but does not compile.
   `-S` compiles the file but does not assemble.
   `-c` compiles and assembles the source file but does not link.
4) **`-E`** only does preprocessing to the file. The input is the C source code as a text file. In this step it does tasks like removing comments or expanding included header files. The output is in the form of preprocessed source code, which is then sent to the standard output.

-S takes the preprocessed file and compiles it into assembler language. The output is in the form of an assembler code file for each specified input file.

-c assembles the source files but does not link it to the appropriate libraries. The output is in the form of an object file for each source input file.

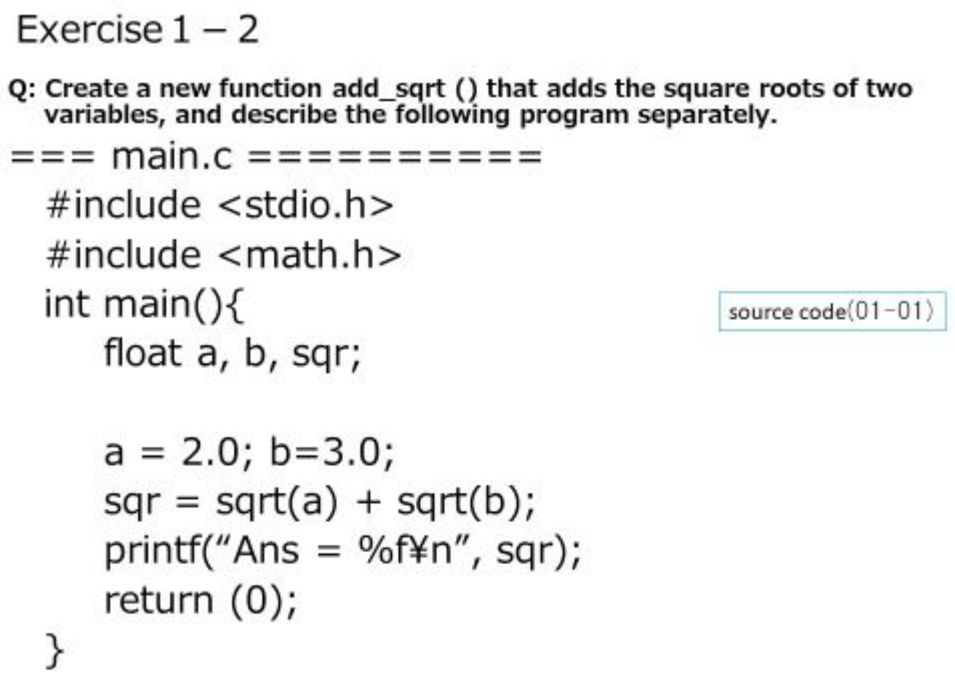
5) **-o** specifies where the primary output should be placed

-O optimizes by trying to reduce code size and execution time.

Reference:

https://gcc.gnu.org/onlinedocs/gcc/Option-Index.html

【Exercise 1-2】



Exercise 1 – 2

Q: Create a new function add_sqrt () that adds the square roots of two variables, and describe the following program separately.

```
=== main.c ==========
  #include <stdio.h>
  #include <math.h>
  int main(){
      float a, b, sqr;

      a = 2.0; b=3.0;
      sqr = sqrt(a) + sqrt(b);
      printf("Ans = %f¥n", sqr);
      return (0);
  }
```

source code(01-01)

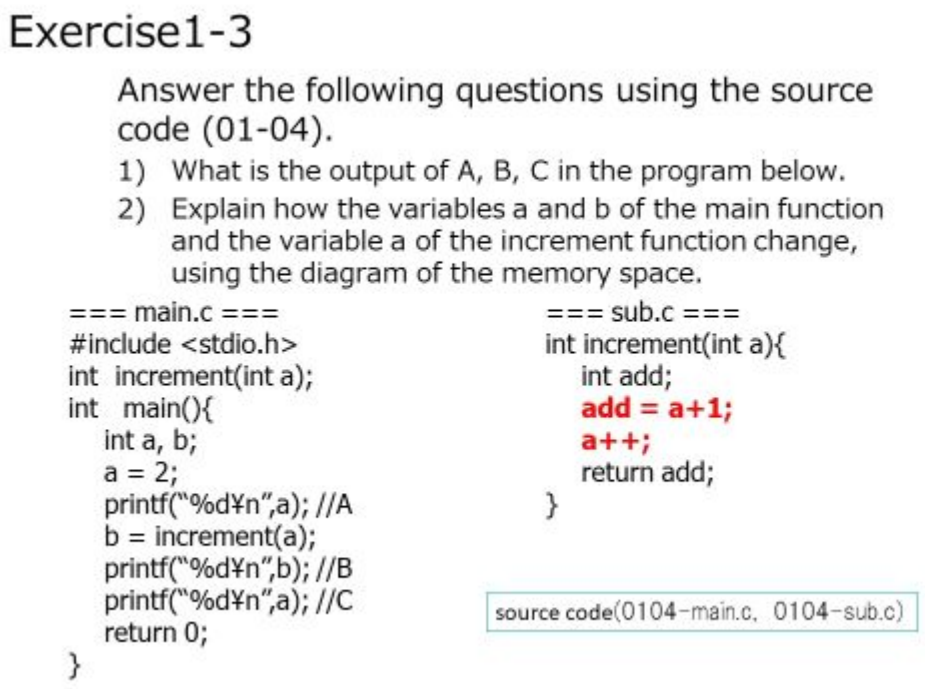
```
float add_sqrt(float a, float b){
    return sqrt(a) + sqrt(b);
}
```

I am not entirely sure about what I should “describe” about the program here.

The original program initiates two variables and find the sum of their square roots by saving it into a third variable.

By creating a separate function `add_sqrt(float a, float b)` outside of `main()`, `add_sqrt()` can be called within `main()`, and the two variables in `main()` can be passed through as arguments.

【Exercise 1-3】



Exercise1-3

Answer the following questions using the source code (01-04).

1) What is the output of A, B, C in the program below.
2) Explain how the variables a and b of the main function and the variable a of the increment function change, using the diagram of the memory space.

```
=== main.c ===
#include <stdio.h>
int  increment(int a);
int   main(){
    int a, b;
    a = 2;
    printf("%d¥n",a); //A
    b = increment(a);
    printf("%d¥n",b); //B
    printf("%d¥n",a); //C
    return 0;
}
```

```
=== sub.c ===
int increment(int a){
    int add;
    add = a+1;
    a++;
    return add;
}
```

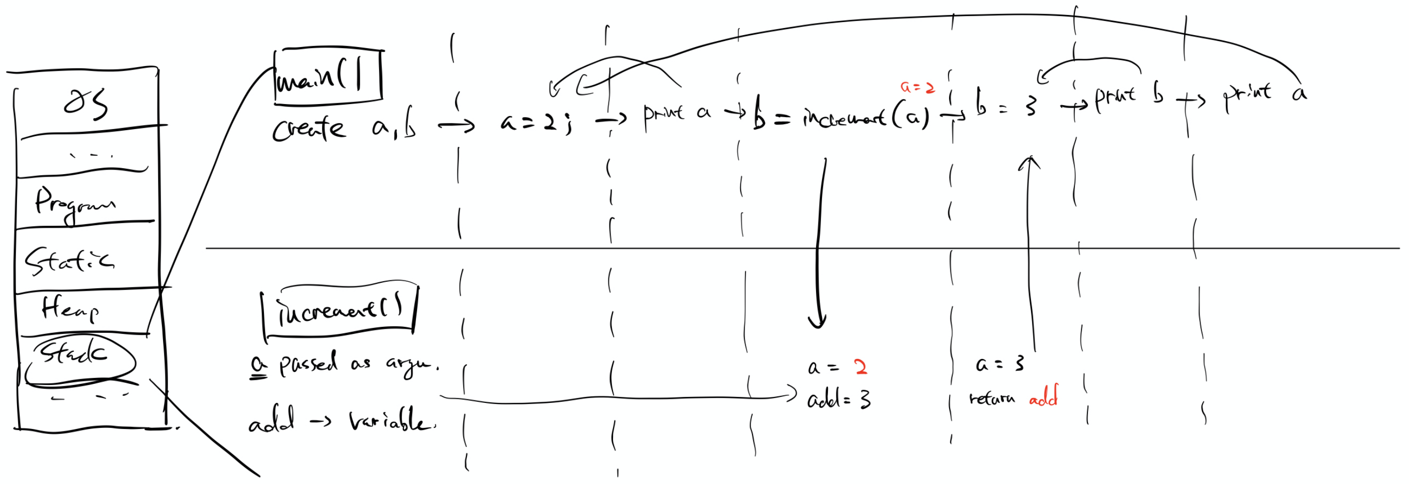
source code(0104-main.c, 0104-sub.c)

1) At A, it should print 2;

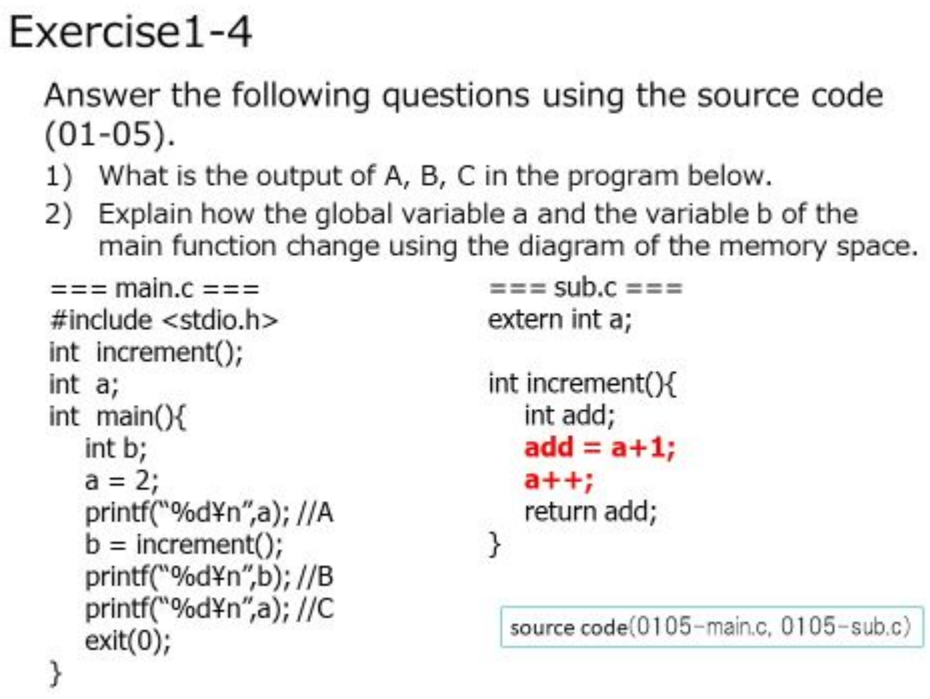
   At B, it should print 3;

   At C, it should print 2.

2)



【Exercise 1-4】



Exercise1-4

Answer the following questions using the source code (01-05).

1) What is the output of A, B, C in the program below.
2) Explain how the global variable a and the variable b of the main function change using the diagram of the memory space.

```
=== main.c ===
#include <stdio.h>
int  increment();
int  a;
int  main(){
    int b;
    a = 2;
    printf("%d¥n",a); //A
    b = increment();
    printf("%d¥n",b); //B
    printf("%d¥n",a); //C
    exit(0);
}
```

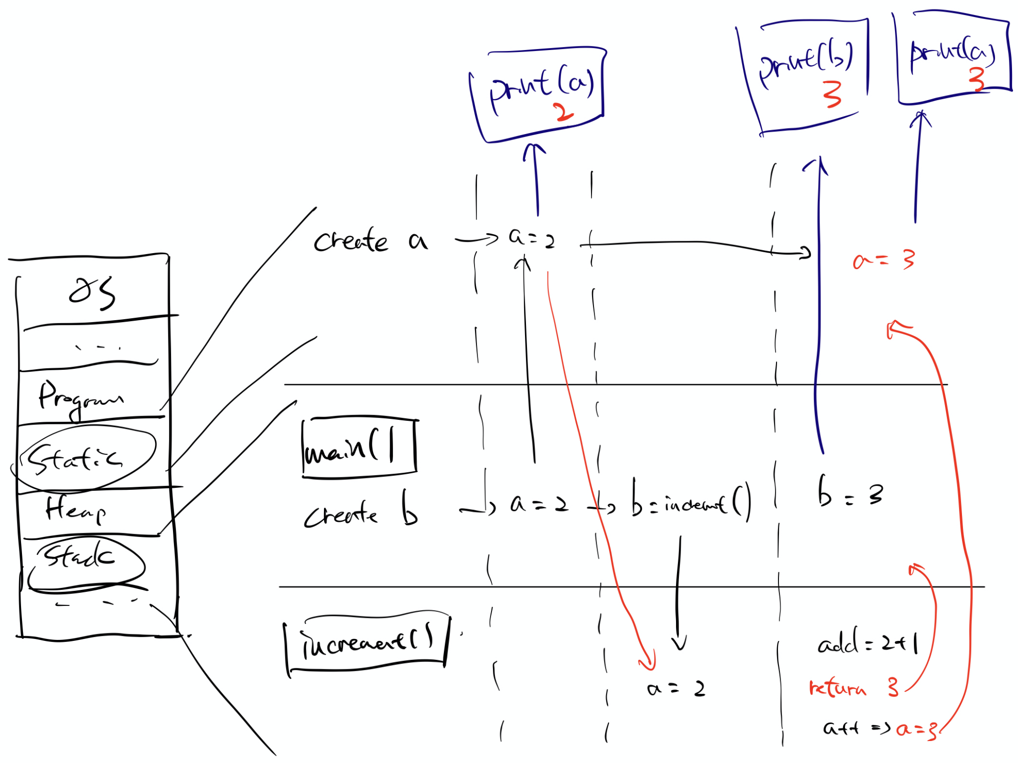
```
=== sub.c ===
extern int a;

int increment(){
    int add;
    add = a+1;
    a++;
    return add;
}
```

source code(0105-main.c, 0105-sub.c)

1) A prints 2, B prints 3, C prints 3.

2)



OS
. . .
Program
Static
Heap
Stack
create a
a = 2
print(a)
2
print(b)
3
print(a)
3
a = 3
main()
create b
a = 2
b = increment()
b = 3
increment()
a = 2
add = 2+1
return 3
a++ => a=3