2020　College of Engineering Systems
Introduction to programming B　【exercise-week02】

Student ID（202018021）　Name（Fu, Ziyu）

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

【Exercise 2-1】

Exercise 2 – 1

Answer the following questions using the source code (02-01).

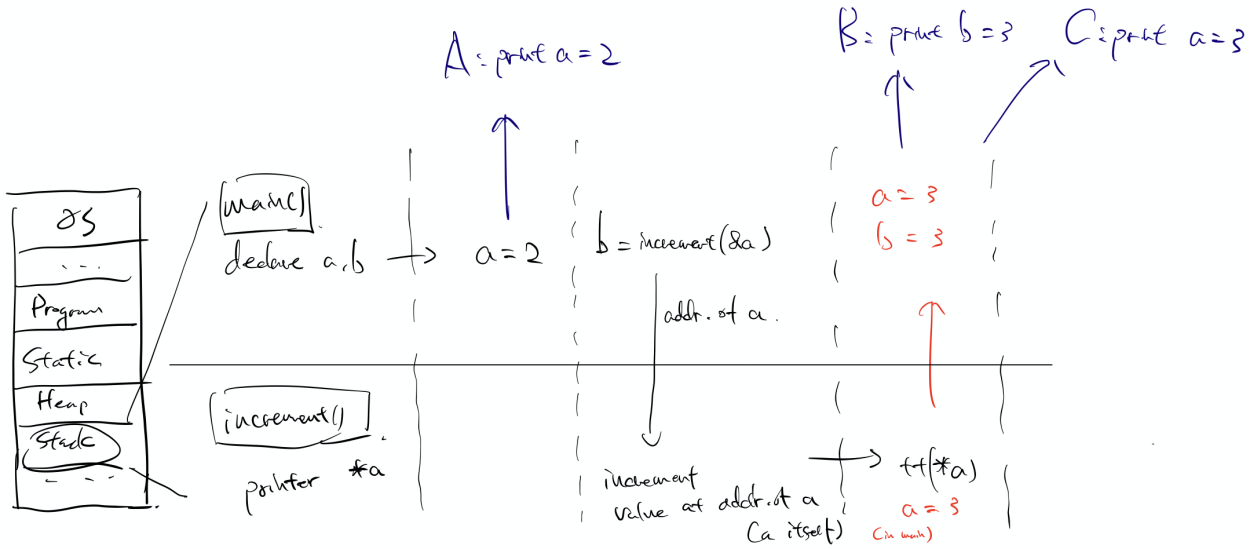
1) What is the output of A, B, C in the program below.
2) Explain how the variables a and b of the main function change and the state of the address "&a" using the diagram of the memory space.

```
=== 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("b=%d¥n",b);  //B
    printf("a=%d¥n",a);  //C
}
```

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

source code(0201-main.c, 0201-sub.c)

1) A = 2; B = 3; C = 3.
2)



【Exercise 2-2】

Exercise 2−2

[Aim] Experience that addresses are mere numerical values and change with each execution (the storage location in memory changes).

1) Run the source code (03-01) 5 times and record the output.
2) Explain the reason why the output result of 1) varies.

1) Console output:

```
The value 1 is stored at addr 4002821676.
The value 1 is stored at addr 3886523948.
The value 1 is stored at addr 4014183980.
The value 1 is stored at addr 3831932460.
The value 1 is stored at addr 3838178860.
```

2) The variable is just being stored at an arbitrary location in the currently available memory on the computer. It can, and most likely will change with every execution of the program.

【Exercise 2-3】

Exercise 2−3

【Aim】

- Understand array declarations and base addresses.
- Understand how the array is stored in memory.

1) Rewrite the "????" part of the source code (03-02) so that the base address is output.
2) Observe the output result of the address of each element of the array, and answer how the address changes when the number of elements increases by one.
3) Explain how the amount of change in 2) is determined.

1) `????` should be `alpha`.
2) The address increments by 4.
3) I am running this on a 64-bit Intel MacBook with the 32-bit kernel in macOS enabled so the default `int` on my computer is 32 bits (4 bytes).
   4 bytes / 1 byte = 4.

【Exercise 2-4】

Exercise 2–4

The following source code (03-03) line 7 "Access to array", Rewrite the expression using pointers.。

```
main()
{
   int  i=0;
   static int  alpha[]={1,2,3,4,5};
   while (i<5)
   {
        printf("The value in %u is %d ¥n", &alpha[i],
   alpha[i]);
        i++;
   }
}
```

The red part is equivalent to **`*(alpha + i)`**.

【Exercise 2-5】

Exercise 2−5 : Manipulation of character strings
Answer the output of the following program (03-04).

```
=== main.c =====
int   main()
{
  char *ptr1, *ptr2="kj2";
  ptr1 = ptr2;
  while(*ptr2 != '¥0')
      putchar(*ptr2++);
  while(--ptr2 >= ptr1)
      putchar(*ptr2);
  putchar('¥n');
}
```

【 String declarative method 】
char *ptr2 ="kj2";

Console output:

kj22jk

The program outputs the value of **ptr2** in the forward direction until **NULL** pointer, and then traces the address backwards until the first address bit in **ptr1** (which is the exact same pointer value as **ptr2**).

【Exercise 2-6】

Exercise 2−6：If "increment(&a)" in the main function of source code (03-05) is "increment(a)",

1) Consider what kind of processing is performed in the increment function.
2) Answer what the program will do.

```
=== main.c =====
#include <stdio.h>

int  increment(int *a);

int   main()
{
   int a;
   a=0;
   increment(&a);
   printf("a=%d¥n",a);
}
```

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

1) The **`increment()`** function takes a pointer variable (address) and increments the value of that variable at the given address.
2) It will increment the a in **`main()`** by 1. In other words, the program will output "**`a=1`**".

【Exercise 2-7】

Exercise 2−7

Based on the source code (02-02), implement the process of masking decimal values using type conversion (cast).

```c
# include <stdio.h>

int main (){
  double v1, v3;
  int v2;

  printf("Input a float number: ");
  scanf("%lf",&v1);

  v2 = (int) v1;
  v3 = v1 - v2;

  printf("%lf：正数部%d, 小数部%f\n",v1,v2,v3);
}
```

Sample output:

Input a float number: 3.2341345
3.234135：正数部 3, 小数部 0.234135