C Programming

Tutorial 5

1. Do you see any problems from the following statements?

```
char a1[] = "ABCD", a2[] = "abcdef", a3[] = "12345";
strcpy(a1, a2);
strcat(a2, a3);
```

a1 and a2 does not have enough space.

2. Do you think the following code would work?

```
char *p;
strcpy(p, "123");
```

The memory pointed to by p is NOT allocated yet.

3. Do you see any problem with the following code?

```
char *p;
*p = malloc(8);
```

```
p = malloc(8);
```

4. When you need a large amount of memory for, e.g., an array, what issues do you need to consider?

Use dynamic memory allocation to have control on when to request and release the memory. Avoid requesting a large amount of continuous memory space.

5. When you freed the memory pointed to by p, what would be the value of p?

The value of p is unchanged. But, it is a good practice to set p to NULL. In theory, after free p is no longer a valid pointer anymore.

6. In function func(), you have the following code:

…

int *p;

p = malloc(80);

…

Since p is local to func(), do you really need to do free(p)?

p is local, but the memory pointed by p is static. So you need to do free(p).

7. Give a statement, which asks for memory space for an array of n ints using malloc().

int *a = malloc(n * sizeof(*a));

8. The following statement is correct. But, do you see any potential problems with it?

ptr = realloc(ptr, 20 * sizeof(int));

If not successful, NULL will be returned to ptr. Use a tmp pointer instead of ptr to avoid this.