C Programming

Tutorial 3

1. Declare p.

a. p is an array of n pointers to int

```
int *p[n];
```

b. p is an array of n pointers to functions that return an int.

```
int (*p[n])();
```

c. p is an array of n pointers to functions that return pointers to int.

```
int *(*p[n])();
```

d. p is an array of n pointers to functions that return pointers to functions that return an int.

```
int (*(*p[n])())();
```

e. p is an array of n pointers to functions that return pointers to functions that return pointers to int.

```
int *(*(*p[n])())();
```

2. How are the following statements related to each other?

```
char a[] = “read-only memory?”;

char *pa = a;

char *pb = “read-only memory?”;

char *pc = “read-only memory?”;
```

Can you assign a new value to a[0], pa[0], pb[0], or pc[0], for example?

It is system dependent. pb and pc may point to the same char in the same block of read-only memory.

3. Using the following example, discuss the difference between a pointer to an array and a pointer to the first element of an array.

```
char weekday[10] = “Mon”;

char week[7][10] = {“Mon”, “Tue”, ...};

char *p;

char (*pw)[10];
```

Which of the following are correct statements?

The key point is to check the types on both sides – do they match.

```
p = weekday; p++;
```
Correct.

```
p = &weekday;
```
Wrong.

```
p = week;
```
Wrong.

```
p = &week;
```
Wrong

```
pw = weekday;
```
Wrong

```
pw = &weekday; pw++;
```
Correct statements. BUT, pw++ not safe.

```
pw = week; pw++;
```
Correct.

```
pw = &week;
```
Wrong.

4. What is the output of printf?

```
int array[5], i, *ip;

for(i = 0; i < 5; i++) array[i] = i;

ip = array;

printf("%d\n", *(ip + 3 * sizeof(int)));
```

This may be what you wanted: printf("%d\n", *(ip + 3));

5. Any thing wrong with the following functions calls?

int r1, r2, (*fp)(), func();

fp = func;

r1 = (*fp)();

r2 = fp();

Both calls are correct.

6. I have an array and an integer variable: int num_element, array[] = {0, 1, 2, 3, 4}; Can you write a statement which tells the number of elements in array? Can you write a standalone function to do this job?

num_element = sizeof(array)/sizeof(array[0]);

7. We have the following declaration: int a[4][4], (*b)[4], *c[4], **d; Since an array name is usually converted to a pointer, it seems reasonable if we pass a, b, c or d to a function expecting a pointer to pointer (e.g., int func(int **);). Do you agree?

We can only pass c and d. a and b can be passed as pointer to array, but NOT pointer to pointer.