C Programming

Tutorial 6

1. We created a list of linked nodes using malloc(). When you do free(ptrnode1), where ptrnode1 is the pointer to the first node in the list, do you free the memory for the whole list or only for the first node?

Only the first node. malloc is a low level function.

2. When you do structure assignment, is there any thing that potentially worries you? (for example, a pointer element is involved)

In general, no – if you want to assign one struct variable to another of the same type. But, if you want to assign values directly to the whole structure, this is not allowed. You can only do this for initialisation (therefore it is called initialisation, not assignment).

However, keep in mind that it does not do deep copying.

3. I used the statement (int *p = malloc(5);) in two programs. It worked for one but not for the other. What could be the problem?

Perhaps, the one that did not work was declared as a static or non-local variable. Function calls are not allowed in such situations.

4. Do you see any problem with this statement: char *p = malloc(strlen(s));?

It is a subtle but fatal mistake. The size should be strlen(s)+1.

5. Any thing wrong with the following statement, trying to dynamically allocate an array?

```
char *char_array = malloc(num_chars);
int *int_array = malloc(num_ints);
```

int *array = malloc(sizeof(*array));

6. Is the free'ed memory returned to the OS?

No. it is still reserved for future dynamic memory allocation. Program memory will generally be returned to the OS upon the execution of the program.

7. See the following code. What is wrong with it? How would you fix the problem?

```
char *func(int n)
{
        char a[10];
        … /* do something about a */
        return a;
}


char *func(int n)
{
        char *a = malloc(10*sizeof(*a);
        … /* do something about a */
        return a;
}

char array[10]; /* global variable */
/* or local variable: char *array = malloc(10*sizeof(*array); */
func(n, array);
…
char *func(int n, char *a)
{
        … /* do something about a */
        return a;
}

char *func(int n)
{
        static char a[10];
        … /* do something about a */
        return a;
}
```

8. See the following statement. Gives the possible returns that the statement could give.

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

Can I do int *tmp = realloc(ptr, 10 * sizeof(int)); int *tmp = realloc(ptr, 0);

Check lecture notes for details.

Case 1: request failed, returns NULL.

Case 2: if there is enough contiguous space right after the original allocation, expand to the new size and returns ptr.

Case 3: if there is no enough contiguous space, allocation the space from a different location and returns the new base address.

You can do int *tmp = realloc(ptr, 10 * sizeof(int)); You got lucky, if int *tmp = realloc(ptr, 0); worked for you, but do not rely on it.

9. Do you really want to free the memory just before finishing the execution of the program?

Yes. good practice cleaning up before you go. The program may be later integrated into a large program.

No?

10. Discuss how to create dynamic arrays, one-dimensional, two-dimensional, three-dimensional, …. Are they really arrays?

int *a, **a, ***a;