Kayley Seow

Chapter 7

Pages 340 – 343

R7.1, 2, 4, 7, 8, 15, 16, 19, 21, 22, 23, 24, 25, 26, 27

Chapter 7 Written Homework

R7.1 Trace the following code. Assume that a and b are stored at 20300 and 20308. Your trace table should have entries for a, b, and p

```
double a = 1000;
double b = 2000;
double* p = &a;
*p = 3000;
p = &b;
a = *p * 2;
```

Step	**Statement**	**Notes**	**a**	**b**	**p**	**p***
1	Original Code		1000	2000	20300	
2	*p = 3000;	Assigns *p with 3000	1000	2000	20300	3000
3	p = &b;	Assign p with b's address	1000	2000	20308	3000
4	a = *p * 2;	Assigns a with *p * 2	2000*2, 4000	2000	20308	3000

R7.2 Trace the following code. Assume that a and b are stored at 20300 and 20308. Your trace table should have entries for a, b, p, and q.

```
double a = 1000;
double b = 2000;
double* p = &a;
double* q = &b;
*p = *q;
p = q;
*p = 3000;
```

Step	**Statement**	**Notes**	**a**	**b**	**p**	**q**
1	Original		1000	2000		
2	double* p = &a;	Assigns p with a's address	1000	2000	20300	
3	double* q = &b;	Assigns q with b's address	1000	2000	20300	20308

4	*p = *q;	Assigns a with b's value	2000	2000	20300	20308
5	p = q;	Assigns p to q, pointers change too	2000	2000	20308	20308
6	*p = 3000;	Assigns the thing p was pointing to a value of 3000	2000	3000	20308	20308

R7.4 Explain the mistakes in the following code. Not all lines contain mistakes. Each line depends on the lines preceding it.

```
1  double a = 1000;
2  double* p = a;
3  int* p = &a; //a is a double, not an int
4  double* q;
5  *q = 2000; //points to memory, 2000 not stored in memory
6  int* r = NULL;
7  *r = 3000; //points to memory, 3000 not stored in memory
```

R7.7 Suppose the array primes, defined as double primes[] = { 2, 3, 5, 7, 11, 13 }; starts at memory location 20300. What are the values of:

a. primes
   20300
b. *primes
   2
c. primes + 4
   20332
d. *(primes + 4)
   11
e. primes[4]
   11
f. &primes[4]
   20332

R7.8 Suppose the array primes is defined as double primes[] = { 2, 3, 5, 7, 11, 13 }; Consider the sum function discussed in Section 7.2.3. What are the values of:

a. sum(primes, 6);
   41
b. sum(primes, 4);
   17
c. sum(primes + 2, 4);
   36
d. sum(primes, 0);

0

e. sum(NULL, 4);
   A compile error.

R7.15 What is the difference between the following three variable definitions?

a. char* p = NULL;
   p is a pointer to a char, pointing to NULL
b. char* q = "";
   q is a pointer to a char, pointing to an empty string
c. char r[1] = { '\0' };
   r is an array of size 1, string terminator character

R7.16 Consider this program segment:

```
char a[] = "Mary had a little lamb";
char* p = a;
int count = 0;
while (*p != '\0')
{
        count++;
        while (*p != ' ' && *p != '\0') { p++; }
        while (*p == ' ') { p++; }
}
```

What is the value of count at the end of the outer while loop?

count = 5

R7.19 What happens if you forget to delete an object that you obtained from the heap? What happens if you delete it twice?

If you forget to delete the object, you will free your memory. If you delete it twice, you will get an error.

R7.21 Find the mistakes in the following code. Not all lines contain mistakes. Each line depends on the lines preceding it. Watch out for uninitialized pointers, NULL pointers, pointers to deleted objects, and confusing pointers with objects.

```
1   int* p = new int;
2   p = 5; //is a pointer, can't assign like this
3   *p = *p + 5; //p not properly initialized
4   string s = "Harry";
5   *s = "Sally"; //not valid, s is a string
6   delete &s;
7   int* a = new int[10];
8   *a = 5;
9   a[10] = 5; //only 0 to 9 is valid
10    delete a;
11    int* q;
12    *q = 5; //not initialized properly
```

```
13    q = p; //q not pointing to memory
14    delete q;
15    delete p;
```

R7.22 How do you define a triangular two-dimensional array using just vectors, not arrays or pointers?

A vector is unlike the arrays, which are static. In this case, we use the push back function in order to enter the elements into the vector.

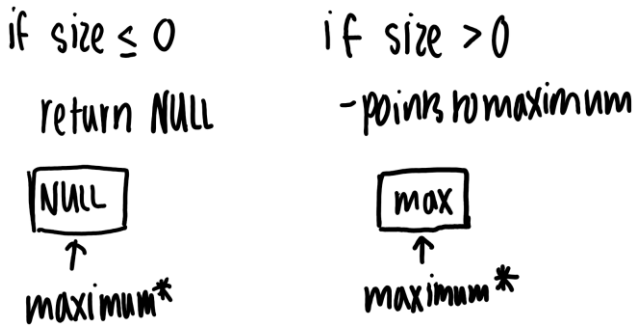
R7.23 Rewrite the statements in Section 7.7.3 so that the street address and employee structures are allocated on the heap.

```
struct StreetAddress *accounting = new StreetAddress;
accounting->house_number = "1729";
accounting->street_name = "Park Avenue";
```

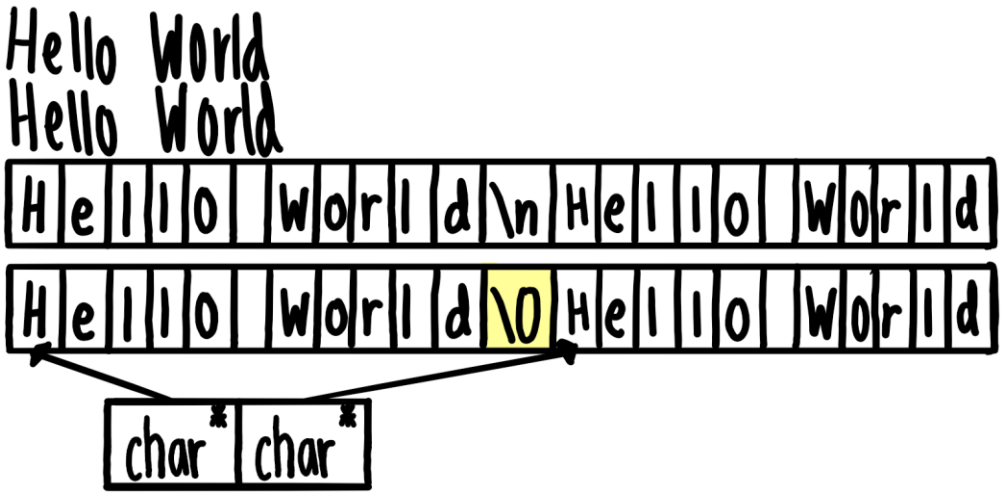
R7.24 Design a structure type Person that contains the name of a person and pointers to the person's father and mother. Write statements that define a structure value for yourself and your parents, correctly establishing the pointer links. (Use NULL for your parents' parents.)

```
struct Person
{
        string name;
        Person *dad;
        Person *mom;
}
int main() {
        struct Person*me = new struct Person;
        struct Person*mydad = new struct Person;
        struct Person*mymom = new struct Person;
        me -> dad = mydad;
        me -> mom = mymom;
        me -> name = "Kayley";
        me -> dad -> name = "HS";
        me -> mom  -> name = "MH";
        me -> dad -> dad = NULL;
        me -> dad -> mom = NULL;
        me -> mom -> dad = NULL;
        me -> mom -> mom = NULL;
        return 0;
}
```

R7.25 Draw a figure showing the result of a call to the maximum function in Exercise P7.4.



R7.26 Draw a figure showing the pointers in the lines array in Exercise P7.13 after reading a few lines.



R7.27 Section 7.6 described an arrangement where each item had a pointer to the user who had checked out the item. This makes it difficult to find out the items that a particular user checked out. To solve this problem, have an array of strings user_names and a parallel array loaned_items. loaned_items[i] points to an array of char* pointers, each of which is a name of an item that the ith user checked out. If the ith user didn't check out any items, then loaned_items[i] is NULL. Draw a picture of this arrangement.