B10632026 四資工三甲 吳苡瑄

1. Exercise 7.9 (d) and (e).

Assume the following rules of associativity and precedence of expressions:

Precedence:

Highest	*, /, not
	+, - , &, mod
	-(unary)
	=, /=, <, <=, >=, >
	and
Lowest	or, xor

Associativity: Left to right

Show the order of evaluation of the following expressions by parenthesizing all subexpressions and placing a superscript on the right parenthesis to indicate order. For example. for the expression : `a + b * c + d`

The order of evaluation would be represented as: `((a + (b * c)1)2 + d)3`

(d) -a or c = d and e

Ans:

`((-a)1 or ((c = d)2 and e)3)4`

(e) a > b xor c or d <= 17

Ans:

`(((a > b)1 xor c)3 or (d <= 17)2)4`

2. Exercise 7.10 (d) and (e).

Show the order of evaluation of the expressions of Problem 9, assuming that there are no precedence rules and all operators associate right to left.

(d) -a or c = d and e

Ans:

`(-(a or (c = (d and e)1)2)3)4`

(e) a > b xor c or d <= 17

Ans:

`(a > (b xor (c or (d <= 17)1)2)3)4`

3. Exercise 7.13.

Let the function `fun` be defined as

```
int fun(int *k) {
    *k += 4;
    return 3 * (*k) - 1;
}
```

Suppose `fun` is used in a program as follows:

```
void main() {
    int i = 10, j = 10, sum1, sum2;
    sum1 = (i / 2) + fun(&i);
    sum2 = fun(&j) + (j / 2);
}
```

What are the values of `sum1` and `sum2`

a. operands in the expressions are evaluated left to right?

Ans:

```
sum1 = (10 / 2) + (3 * (10 + 4) - 1)
     = 5 + (3 * 14 – 1)
     = 5 + (42 – 1)
     = 46
sum2 = (3 * (10 + 4) - 1) + (14 / 2)
     = (3 * 14 – 1) + 7
     = 41 + 7
     = 48
```

b. operands in the expressions are evaluated right to left?

Ans:

```
sum1 = (14 / 2) + (3 * (10 + 4) - 1)
     = 7 + (3 * 14 – 1)
     = 7 + (42 – 1)
     = 48
sum2 = (3 * (10 + 4) - 1) + (10 / 2)
     = (3 * 14 – 1) + 5
     = 41 + 5
     = 46
```

4. Exercise 7.19.

Consider the following C program:

```
int fun (int *i) {
    *i += 5;
    return 4;
}
void main() {
    int x = 3;
    x = x + fun (&x);
}
```

What is the value of x after the assignment statement in main, assuming

a. operands are evaluated left to right.

Ans:

```
x = 3 + 4
  = 7
```

b. operands are evaluated right to left.

Ans:

```
x = (3 + 5) + 4
  = 12
```

5. Programming Exercise 8.3(a) in C.

Rewrite the following code segment using a multiple-selection statement in C:

```
if ((k == 1) || (k == 2)) j = 2 * k - 1
if ((k == 3) || (k == 5)) j = 3 * k + 1
if (k == 4) j = 4 * k - 1
if ((k == 6) || (k == 7) || (k == 8)) j = k - 2
```

Ans:

```
switch(k) {
    case 1:
    case 2:
        j = 2 * k - 1;
        break;
    case 3:
    case 5:
        j = 3 * k + 1;
        break;
    case 4:
        j = 4 * k - 1;
        break;
    case 6:
    case 7:
    case 8:
        j = k - 2;
        break;
}
```

6. Programming Exercise 8.4 in C.

Consider the following C program segment. Rewrite it without using gotos or breaks.

```
j = -3;
for (i = 0; i < 3; i++) {
    switch (j + 2) {
        case 3:
        case 2: j--; break;
        case 0: j+=2; break;
        default: j = 0;
    }
    if (j > 0) break;
    j = 3 - i;
}
```

Ans:

```
int j = -3;
for (i = 0; i < 3; i++) {
    int compare = j + 2;
    if (compare == 3 || compare == 2) {
        j--;
    }
    else if (compare == 0) {
        j += 2;
    }
    else {
        j = 0;
    }

    if (j > 0) {
        i = 3;
    }
    else {
        j = 3 - i;
    }
}
```

7. Exercise 9.5. Justify your answers.

Consider the following program written in C syntax:

```
void swap (int a, int b) {
    int temp;
    temp = a;
    a = b;
    b = temp;
}
void main () {
    int value = 2, list[5] = {1, 3, 5, 7, 9};
```

```
        swap (value, list[0]);
        swap (list[0], list[1]);
        swap (value, list[value]);
    }
```

For each of the following parameter-passing methods, what are all of the values of the variables `value` and `list` after each of the three calls to swap?

a. Passed by value

Ans:

Pass by value only copy the value into the function, no value will be changed after operation of the function.

```
    swap (value, list[0]);
        value = 2, list = {1, 3, 5, 7, 9}
    swap (list[0], list[1]);
        value = 2, list = {1, 3, 5, 7, 9}
    swap (value, list[value]);
        value = 2, list = {1, 3, 5, 7, 9}
```

b. Passed by reference

Ans:

What do in the function will effect the value of that address, so values will be changed after the operation of the function.

```
    swap (value, list[0]);
        value = 1, list = {2, 3, 5, 7, 9}
    swap (list[0], list[1]);
        value = 1, list = {3, 2, 5, 7, 9}
    swap (value, list[value]);
        value = 5, list = {3, 2, 2, 7, 9}
```

c. Passed by value-result

Ans:

After operation, value will be assign back to the parameter, so values will be changed after the operation of the function.

```
    swap (value, list[0]);
        value = 1, list = {2, 3, 5, 7, 9}
    swap (list[0], list[1]);
        value = 1, list = {3, 2, 5, 7, 9}
    swap (value, list[value]);
        value = 5, list = {3, 2, 2, 7, 9}
```

8. Exercise 9.7. Justify your answers.

Consider the following program written in C syntax:

```
void fun (int first, int second) {
    first += first;
    second += second;
}
int main () {
    int list [2] = {1, 3};
    fun (list[0],list[1]);
}
```

For each of the following parameter passing methods, what are the values `list` array after execution?

a. Passed by value

Ans:

Pass by value only copy the value into the function, no value will be changed after operation of the function.

first = 1, second = 3;

b. Passed by reference

Ans:

What do in the function will effect the value of that address, so values will be changed after the operation of the function.

first = 2, second = 6;

c. Passed by value-result

Ans:

After operation, value will be assign back to the parameter, so values will be changed after the operation of the function.

first = 2, second = 6;