COMP20005 Workshop Week 3

0	**Q&A**: expressions, **`if`** statement and **`for`** loop
1	Discuss Exercise **`3.02`**
2	Design **`3.06`**, 3.07. Note: How **`scanf`** executed.
	5-min break
LAB	implement **`Ex 3.06`** using incremental development [Time permitting] Design and implement a solution to **`Ex 3.07b`**. [Extra] Implement **`Ex 4.03`**, **`Ex 4.04`**
Past & Future	**Lectures W2:** expression, **`if`** & **`switch`**, **`for`** and nested **`for`** loop **Quiz 1** (Week 6): *likely* covers all stuffs learned in weeks 1—5

if . . .

```
if ( condition_is_true ) {
   // do some stuffs A1
}
// continuing ...
```

```
if ( guard ) {
   // do some stuffs A1
} else {
   // do alternatives A2
}
// continuing ...
```

```
int a= 0, b= 0;

if (scanf("%d%d", &a, &b) != 1) {
   printf("Invalid input\n");
   exit(EXIT_FAILURE);
}

// print out the larger of a, b
if ( a > b ) {
   printf("max = %d\n", a);
} else {
   printf("max = %d\n", b);
}
```

fill in for computing the min of a,b,c

```
int main(int argc, char *argv[]) {
   int a, b, c;
   printf("Enter int value for a,b,c: ");
   if (scanf("%d %d %d", &a, &b, &c) != 3) {
      printf ("invalid input, I could not get 3 integers\n");
      exit(EXIT_FAILURE);
   }
   ???

   printf("The min of %d, %d, and %d is %d\n", a, b, c, ?);
   return 0;
}
```

Ex 3.02: use grok to *see and do* this exercise!

Trace the action of these statements, and determine the values printed out by each of the `printf` *statements. Assume that all variables have been declared to be of type* `int`.

Ex 3.2 a): (use grok and/or pens & papers)

```
1  i = 3;  j = 4;

2  if ( i<j && j<6 ) {

3      i= i+j;
4  } else {
5      j= i+j;
6  }
7  printf ("i = %d, j = %d\n", i, j);
```

```
i =  , j =
```

3.2 b)

```
1   i = 3;  j = 4;  k = 7;
2   if ((i<j || j<k) && j<i) {
3       i = i+1;
4       if (i*i>k) {
5           k = k+1;
6       }
7   } else {
8       j = j+1;
9       if (i*i>k) {
10          k = k+2;
11      }
12  }
13  printf (“i = %d, j = %d, k = %d\n”, i, j, k);
```

i = , j = , k =

3.2 d)

```
1  x = 1; y = 2;
2  if (x>y)
3      printf (“x = %d, y = %d\n”, x, y);
4  x = x+1;
5  if (x<y){
6      printf (“x = %d, y = %d\n”, x, y);
7  }
8      y = y+2;
   printf (“x = %d, y = %d\n”, x, y);
```

```
x =  , y =
```

3.2 e)

```
1   x = 1; y = 2;
2   if (x>y) {};
3   {
4       printf (“x = %d, y = %d\n”, x, y);
5       x = x+1;
6   }
7   if (x<y); {
8       printf (“x = %d, y = %d\n”, x, y);
9       y = y+2;
10  }
    printf (“x = %d, y = %d\n”, x, y);
```

```
x =  , y =
???
```

3.2 f)

```
1  x = 0; y = 0;
2  if (y<x) {
3      printf (“y is smaller\n”);
4  } else if (y=x) {
5      printf (“x and y are equal\n”);
6  } else {
7      printf (“y is greater\n”);
8  }
```

???

3.2 c)

```
1  month = 7;
2  if (month == 2) {
3      days = 28;
4  } else if (month == 4 || month==6 ||
5  month==9 || 11) {
6      days = 30;
7  } else {
8      days = 31;
9  }
   printf (“days = %d\n”, days);
```

```
days =
```

operator precedence. Example: grok Ex 3.01

++, -- !, -, (*type*)	unary
*, /, % +, -	(binary) arithmetic
<, >, <=, >= ==, !=	relational
&& ||	(binary) logical
=, +=, *=, etc	(binary) assignments

- Skim `Ex 3.01`
- Do Together: `Ex 3.07`

loops

- *Hey, learner, run 10 rounds before getting out!*
- *Sir, ...*

“program” Anh-a-Computer-Driver to drive 10 rounds

Line	Hey Anh, do:
0	START
1	run_a_round;
2	run_a_round;
3	run_a_round;
4	run_a_round;
5	run_a_round;
6	run_a_round;
7	run_a_round;
8	run_a_round;
9	run_a_round;
10	run_a_round;
11	END

Line	Hey Anh, do:
0	START
1	
2	END
3	
4	
5	
6	
7	
8	
9	
10	
11	

The for loop

init *guard* *update*

```
for ( count= 0 ; count < 5 ; count++ ) {
      run_a_lap;
}
```

start of **loop body**

loop body

end of **loop body**

```
// another example
for ( r= 0;  r<5 && petrol_OK ;  r++ ) {
      run_a_lap;
}
```

Examples: build code fragments for computing:

Use the space of `grok Ex 4.04` and do with Anh

S= sum of all real values from input

Then, do together: `Ex 4.04`

while loop: loop without explicit *init* and *update*

```
sum=0;
for (     ;  scanf("%lf",&x)==1  ;     ) {
    sum += x;
}
```

```
// equivalent to the top
sum=0;
while (  scanf("%lf",&x)==1  ) {
    sum += x;
}
```

Compare:

	a= 5; b= 5;	a= b= 5; Assignment is an expression!

	a= a * b;
	n= n+1;
	n += 1;
	m= m-1;
		a *= b;
		n++;
		m--;

	scanf("%d%d", &a, &b);
	
	//rest of the program
		if (scantf("%d%d", &a, &b) != 2) {
		  printf("invalid input\n");
		  exit(EXIT_FAILURE);
		}
		// rest of the program

5-min break

Quiz 1

What **XXX** *should be in the following fragment:*

```
printf("Enter value for a and b : ");
if ( scanf("%d%d",&a,&b) XXX ) {
  printf("Please enter 2 integers\n");
  exit( EXIT_FAILURE );
}
```

A: `!= 0`	**B:** `!= 2`
C: `== 1`	**D:** `== 2`

Quiz 2

If we execute the following fragment:

```
int i, n=0;  char c; float x;
n= scanf("%d%c%f", &i, &c, &x);
```

with the input stream (data from keyboard) of:

```
100.1A200.2
```

the value of `n`, `i`, `c`, *and* `x` *become respectively:*

A: `0 100  A  200.2`	**B:** `3 100  A  200.2`
C: `3 100  .  1`	**D:** `(something else)`

Quiz 3

What is the output of the following fragment:

```
int a=1, b=2;
if ( a = b ) {
  printf("a= %d ", a);
} else {
  printf("b= %d", b);
}
printf("\n");
```

A: a= 1 b= 2	**B:** a= 1
C: a= 2	**D:** b= 2

Quiz 4

What is the output of the following fragment:

```
int i, sum;
for ( i=0, sum=0; i < 10; i += 2 ) {
  sum += i;
  i++;
}
printf("i= %d, sum= %d\n");
```

A: `i= 12, sum= 18`	**B:** `i= 10, sum= 18`
C: `i= 12, sum= 30`	**D:** <something else>

Ex 3.6 (Discuss Design, then Implement)

In the past, Australia had coins in denominations of 50c, 20c, 10c, 5c, 2c, and 1c. Write a program that reads an integer amount of cents between 0 and 99 (your program might check for valid input) and print out the coins necessary to make up that amount of money. For example:

```
bash$ ./calculatechange
Enter amount in cents: 41
The coins required to make 41 cents are:
give a 20c coin
give a 20c coin
give a  1c coin
amount remaining: 0c
```

Note: Don't worry if your program seems a bit clumsy, and not terribly general!

Lab: Implement 3.06 and 3.07 and others in grok C03 and C04

3.06: *In the past, Australia had coins in denominations of 50c, 20c, 10c, 5c, 2c, and 1c. Write a program that reads an integer amount of cents between 0 and 99 (your program might check for valid input) and print out the coins necessary to make up that amount of money.*

3.07: *(if not yet done) Extend your "Fahrenheit to Celsius" program by adding in the reverse transformation. For example:*

```
bash$ ./converter
Enter a temperature: 212C
The temperature 212.0C converts to 413.6F
```

Finished? Then:

- *do* **Ex 4.03** *(Fibonacci numbers),* **4.04** *(prints ASCII table if not yet done)*
- *and/or ask Anh for a funny exercise.*

Note: Don't do **Ex 4.05, 4.09***: we'll do next week*

Additional Exercise:
Create A Quiz
(For those who have finished all today's lab works)

Write a program to perform your own quiz with around five questions. A question can require a number as an answer (e.g: what is the next number after 1 2 4 8?) or a selection (e.g. which choice (A, B, C, or D). After each question you should let the users know if they are correct.

And, at the end of the quiz, you should print the percentage of questions the user gets right. See example on the right.

```
Fun time!

What printf("%d\n", 5%2) prints out? 1
Correct!

Who is current Victoria's premier:
A Alistair   B Dan   C Scott   D Anh
? A
Incorrect

How many seasons in "Game of Thrones":
A 7       B  4       C  8     D  6
? C
Correct!

What is the output of:
if (0==1);printf("0=1 "); printf ("haha\n");
A  0=1 haha     B haha    C  <empty>
? haha
Incorrect

Not bad! You got 2 answers right.
Your score is 50%.
```

Wrap Up

operator precedence is as expected: *unary*, *arithmetic*, *relational*, *logic*, *assignment*

```
if ( guard ) { ... } else { ... }
```

guard : non-zero is TRUE, zero is FALSE

```
for ( init ; guard ; update ) {
   //do something in one iteration
}
```

```
while ( guard ) {
   //do something in one iteration
}
```

reading one tuple of input

```
if (scanf(... ) != ...) {
    printf("...\n");
    exit(EXIT_FAILURE);
}
// process the input tuple
```

reading multiple tuples of input

```
while ( scanf(... ) == ... )
{
    // process next input tuple

}
```