CEng 230 Introduction to C Programming

Seyyit Alper SERT

Department of Computer Engineering
2017-2018 Fall

Web Pages
Official Course Page: ceng230.ceng.metu.edu.tr
Learning Management System (LMS): odtuclass.metu.edu.tr
Contact: alper.sert@ceng.metu.edu.tr

selection structures

conditions

if…else statement

switch statement

Condition is an expression that is either

false (represented by 0) or
true (usually represented by 1)

the expression : (**rest_heart_rate > 75)**

evaluates to 1 (true) when rest_heart_ rate is over 75;

evaluates to 0 (false) if rest_heart_rate is not greater than 75.

TABLE 4.3 The && Operator (and)

operand1	operand2	operand1 && operand2
nonzero (true)	nonzero (true)	1 (true)
nonzero (true)	0 (false)	0 (false)
0 (false)	nonzero (true)	0 (false)
0 (false)	0 (false)	0 (false)

TABLE 4.4 The || Operator (or)

operand1	operand2	operand1 || operand2
nonzero (true)	nonzero (true)	1 (true)
nonzero (true)	0 (false)	1 (true)
0 (false)	nonzero (true)	1 (true)
0 (false)	0 (false)	0 (false)

TABLE 4.5 The ! Operator (not)

operand1	!operand1
nonzero (true)	0 (false)
0 (false)	1 (true)

C accepts *any nonzero value as a representation of true.*

Logical operators

Three logical operators

&& (AND)

|| (OR)

! (NOT) logical complement, negation

```
salary < MIN_SALARY || dependents > 5

temperature > 90.0 && humidity > 0.90

n >= 0 && n <= 100

0 <= n && n <= 100

!(0 <= n && n <= 100)
```

TABLE 4.1 Relational and Equality Operators

Operator	Meaning	Type
<	less than	relational
>	greater than	relational
<=	less than or equal to	relational
>=	greater than or equal to	relational
==	equal to	equality
!=	not equal to	equality

x	power	MAX_POW	y	item	MIN_ITEM	mom_or_dad	num	SENTINEL
-5	1024	1024	7	1.5	-999.0	'M'	999	999

TABLE 4.2 Sample Conditions

Operator	Condition	English Meaning	Value
`<=`	`x <= 0`	`x` less than or equal to 0	1 (true)
`<`	`power < MAX_POW`	`power` less than `MAX_POW`	0 (false)
`>=`	`x >= y`	`x` greater than or equal to `y`	0 (false)
`>`	`item > MIN_ITEM`	`item` greater than `MIN_ITEM`	1 (true)
`==`	`mom_or_dad == 'M'`	`mom_or_dad` equal to `'M'`	1 (true)
`!=`	`num != SENTINEL`	`num` not equal to `SENTINEL`	0 (false)

Operator precedence

TABLE 4.6 Operator Precedence

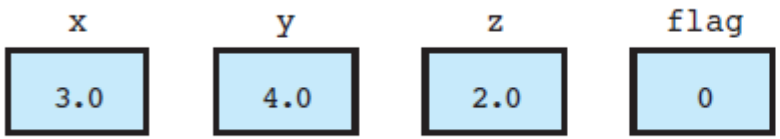
Operator	Precedence
function calls	highest
! + - & (unary operators)	↓
* / %	
+ -	
< <= >= >	
== !=	
&&	
||	
=	lowest

-x - y * z

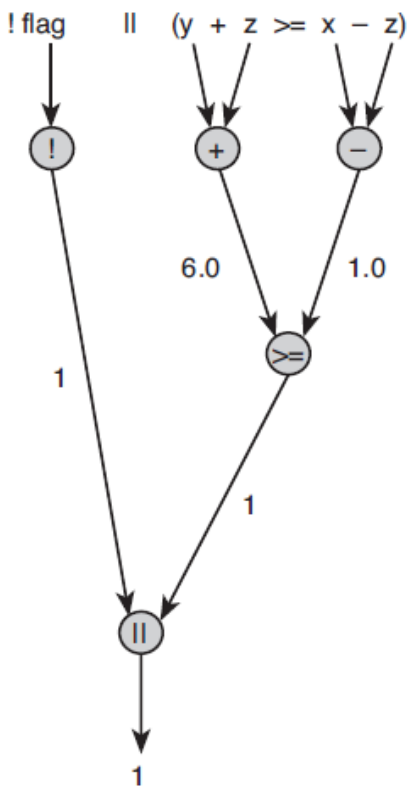
the unary minus is evaluated first (-x), then * , and then the second - .

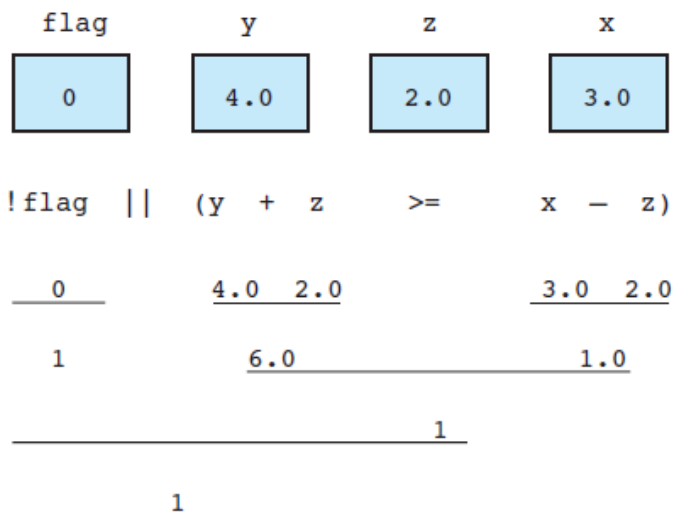
(x < y || x < z) && x > 0.0

in expression x < y || x < z && x > 0.0 first, && is evaluated.




```
1.  !flag                          /* !0 is 1 (true)          */
2.  x + y / z  <=  3.5             /* 5.0 <= 3.5 is 0 (false) */
3.  !flag || (y + z  >=  x - z)    /* 1 || 1 is 1 (true)      */
4.  !(flag || (y + z  >=  x - z))  /* !(0 || 1) is 0 (false)  */
```





short-circuit evaluation

An expression of the form (a || b)
must be true if a is true. Consequently, C stops evaluating the expression when it determines that the value of !flag is 1 (true).

Similarly, an expression of the form (a && b)
must be false if a is false, so C would stop evaluating such an expression if its first operand evaluates to 0.

Writing English Conditions in C

x is 3.0
y is 4.0
z is 2.0

English Condition	Logical Expression	Evaluation
x and y are greater than z	`x > z && y > z`	1 && 1 is 1 (true)
x is equal to 1.0 or 3.0	`x == 1.0 || x == 3.0`	0 || 1 is 1 (true)
x is in the range z to y, inclusive	`z <= x && x <= y`	1 && 1 is 1 (true)
x is outside the range z to y	`!(z <= x && x <= y)` `z > x || x > y`	!(1 && 1) is 0 (false) 0 || 0 is 0 (false)

Comparing Characters

Expression	Value
`'9' >= '0'`	1 (true)
`'a' < 'e'`	1 (true)
`'B' <= 'A'`	0 (false)
`'Z' == 'z'`	0 (false)
`'a' <= ch && ch <= 'z'`	1 (true) if `ch` is a lowercase letter

Logical Assignment

The simplest form of a logical expression in C is a single type **int** value or variable intended to represent the value true or false.

We can use assignment statements to set such variables to true (a nonzero value) or false (0).

```
int in_range, is_letter ;
int n=15;

in_range = (n > -10 && n < 10);
//range check

is_letter = ('A' <= ch && ch <= 'Z') || ('a' <= ch && ch <= 'z');
//checks whether the variable is a letter

int even = (n % 2 == 0);
// assigns 0 (false)  to variable even
```

Complementing a Condition (not a variable)

The condition
status == 'S' && age > 25
is true for a single person over 25.

The complement of this condition is

!(status == 'S' && age > 25**)**

we can write the complement of
age > 25 && (status == 'S' || status == 'D')
as
age <= 25 || (status != 'S' && status != 'D')

The original condition is true for anyone who is over 25, and is either single or divorced.
The complement would be true for anyone who is 25 or younger, or for anyone who is currently married.

Examples

int a = 6 , b = 9 , c = 14 , flag = 1 .

```
c == a + b  ||  !flag
a != 7  &&  flag  ||  c >= 6
!(b <= 12)  &&  a % 2 == 0
!(a > 5  ||  c < a + b)
```

int ans;
int p = 100, q = 50.

ans = (p > 95) + (q < 95);
What is the value of ans?

Complement the expression below

a != 7 && flag || c >= 6

a == 7 || flag && c < 6

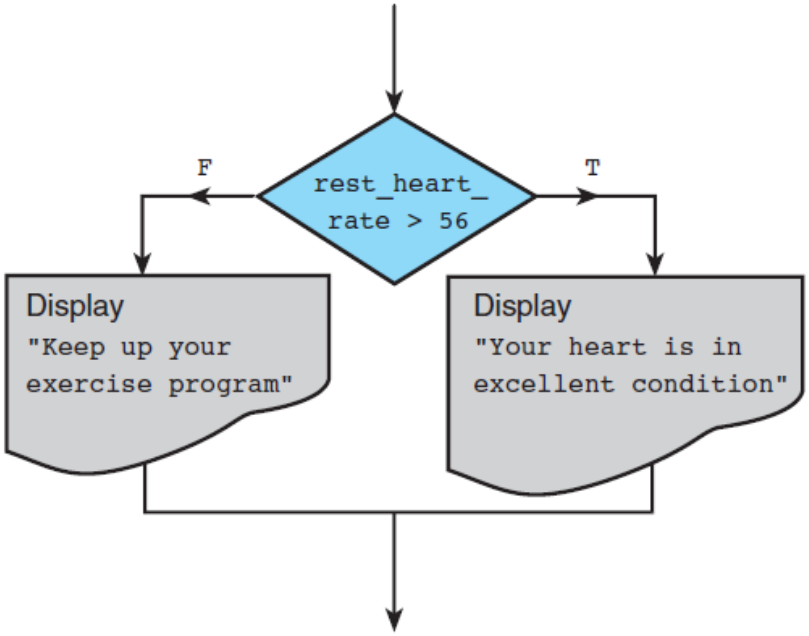
!(1 || 0) 0

!(1 || 1 && 0) 0

!((1 || 0) && 0) 1 (Parenthesis are useful)

if statement

```
if (rest_heart_rate > 56)
      printf("Keep up your exercise program!\n");
else
      printf("Your heart is in excellent health!\n");
```



Flowchart of the **if** statement

```
if (x > 0.0)
pos_prod = pos_prod * x;
```

```
if (crsr_or_frgt == 'C')
  printf("Cruiser\n");
else
  printf("Frigate\n");
```

It displays either Cruiser or Frigate , depending on the character stored in the type char variable crsr_or_frgt .

```
if crsr_or_frgt == 'C'          /* error - missing parentheses */
     printf("Cruiser\n");
printf("Combat ship\n");
```

```
if (crsr_or_frgt == 'C');  /* error - improper placement of ;*/
      printf("Cruiser\n");
printf("Combat ship\n");
```

y= x > 3 ? a+1 : a-1; means

```
İf (x > 3)
        y=a+1;
else
        y=a-1;
```

---

z=(a > b) ? a: b; (finds maximum)

Printf("%d%c", k, (k%10==9) ? 'A' : 'a');

Some exercises

What value is assigned to `x` when `y` is `10.0`?

a.
```
x = 25.0;
if (y != (x - 10.0))
      x = x - 10.0;
else
      x = x / 2.0;
```

b.
```
if (y < 15.0)
      if (y >= 0.0)
            x = 5 * y;
      else
            x = 2 * y;
else
      x = 3 * y;
```

c.
```
if (y < 15.0 && y >= 0.0)
      x = 5 * y;
else
      x = 2 * y;
```

if statements and compound statements

```
if (pop_today > pop_yesterday) {
     growth = pop_today - pop_yesterday;
     growth_pct = 100.0 * growth / pop_yesterday;
     printf("The growth percentage is %.2f\n", growth_pct);
}
```

```
if (ctri <= MAX_SAFE_CTRI) {
      printf("Car #%d: safe\n", auto_id);
      safe = safe + 1;
} else {
      printf("Car #%d: unsafe\n", auto_id);
      unsafe = unsafe + 1;
}
```

Switching values of two variables

```
if (x > y) {                    /* Switch x and y */
      temp = x;                 /* Store old x in temp */
      x = y;                    /* Store old y in x */
      y = temp;                 /* Store old x in y */
}
```

Statement Part	x	y	temp	Effect
	12.5	5.0	?	
`if (x > y) {`				`12.5 > 5.0` is true.
`temp = x;`			12.5	Store old `x` in `temp`.
`x = y;`	5.0			Store old `y` in `x`.
`y = temp;`		12.5		Store old `x` in `y`.

Revise the style of the following `if` statement to improve its readability.

```
if (engine_type == 'J') {printf("Jet engine");
speed_category = 1;}
else{printf("Propellers"); speed_category
= 2;}
```

```
if (engine_type == 'J')
 {
      printf("Jet engine");
      speed_category = 1;
}
else
{
      printf("Propellers");
      speed_category= 2;
}
```

nested if statements and alternative decisions

if statement inside another

```
if (x > 0)
      num_pos = num_pos + 1;
else

      if (x < 0)
            num_neg = num_neg + 1;
      else /* x equals 0 */
            num_zero = num_zero + 1;
```

Comparison of Nested if and Sequence of ifs

```
if (x > 0)
      num_pos = num_pos + 1;
else
      if (x < 0)
            num_neg = num_neg + 1;
      else /* x equals 0 */
            num_zero = num_zero + 1;
```

```
if (x > 0)
      num_pos = num_pos + 1;
if (x < 0)
      num_neg = num_neg + 1;
if (x == 0)
      num_zero = num_zero + 1;
```

Although they do the same thing:

The second solution is less efficient because all three of the conditions are always tested.

In the nested if statement, only the first condition is tested when x is positive.

Multiple-Alternative Decision Form of Nested if

SYNTAX:

```
if (condition_1)
      statement_1
else if (condition_2)
      statement_2
          .
          .
          .
else if (condition_n)
      statement_n
else
      statement_e
```

EXAMPLE:

```
/* increment num_pos, num_neg, or num_zero depending
   on x */
if (x > 0)
      num_pos = num_pos + 1;
else if (x < 0)
      num_neg = num_neg + 1;
else /* x equals 0 */
      num_zero = num_zero + 1;
```

Loudness in Decibels (db)	Perception
50 or lower	quiet
51 – 70	intrusive
71 – 90	annoying
91 – 110	very annoying
above 110	uncomfortable

```
/* Display perception of noise loudness */

if (noise_db <= 50)
      printf("%d-decibel noise is quiet.\n", noise_db);
else if (noise_db <= 70)
      printf("%d-decibel noise is intrusive.\n", noise_db);
else if (noise_db <= 90)
      printf("%d-decibel noise is annoying.\n", noise_db);
else if (noise_db <= 110)
      printf("%d-decibel noise is very annoying.\n", noise_db);
else
      printf("%d-decibel noise is uncomfortable.\n", noise_db);
```

Logic error

```
/* incorrect perception of noise loudness */

if (noise_db <= 110)
      printf("%d-decibel noise is very annoying.\n", noise_db);
else if (noise_db <= 90)

      printf("%d-decibel noise is annoying.\n",
              noise_db);
else if (noise_db <= 70)
      printf("%d-decibel noise is intrusive.\n",
              noise_db);
else if (noise_db <= 50)
      printf("%d-decibel noise is quiet.\n",
              noise_db);
else
      printf("%d-decibel noise is uncomfortable.\n", noise_db);
```

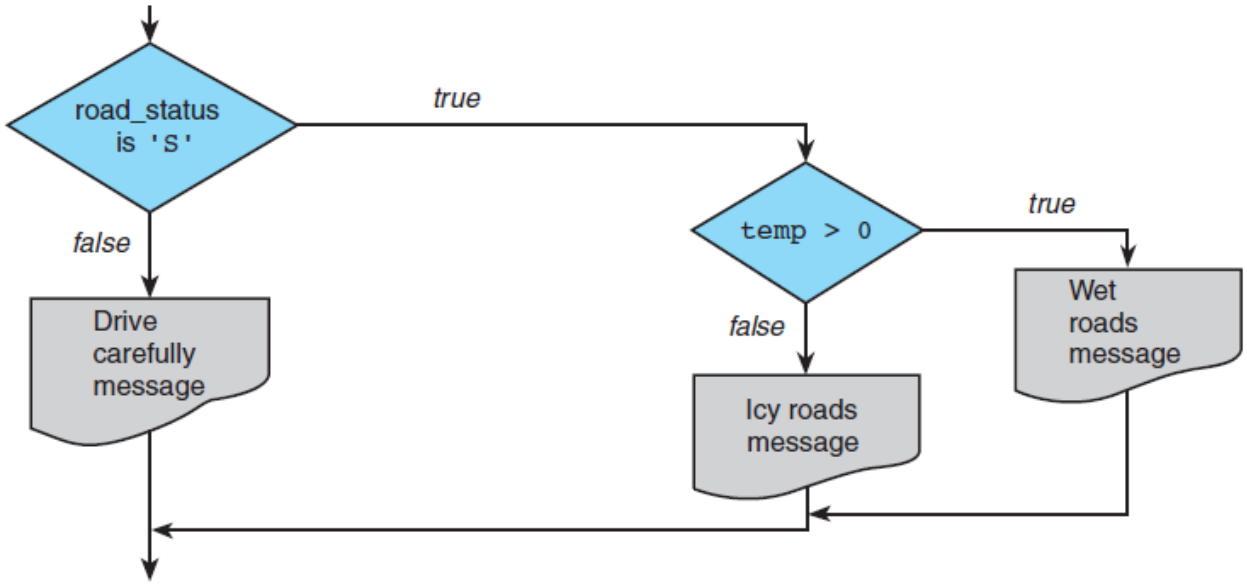
```
* Computes the tax due based on a tax table.



double tax;

if (salary < 0.0)
     tax = -1.0;
else if (salary < 15000.00)                        /* first range        */
     tax = 0.15 * salary;
else if (salary < 30000.00)                        /* second range       */
     tax = (salary - 15000.00) * 0.18 + 2250.00;
else if (salary < 50000.00)                        /* third range        */
     tax = (salary - 30000.00) * 0.22 + 5400.00;
else if (salary < 80000.00)                        /* fourth range       */
     tax = (salary - 50000.00) * 0.27 + 11000.00;
else if (salary <= 150000.00)                      /* fifth range        */
     tax = (salary - 80000.00) * 0.33 + 21600.00;
else
     tax = -1.0;
```

Nested if Statements with More than One Variable

```
/* Print a message if all criteria are met. */
if (marital_status == 'S')
     if (gender == 'M')
          if (age >= 18  &&  age <= 26)
                 printf("All criteria are met.\n");
```

An equivalent statement that uses a single `if` with a compound condition follows.

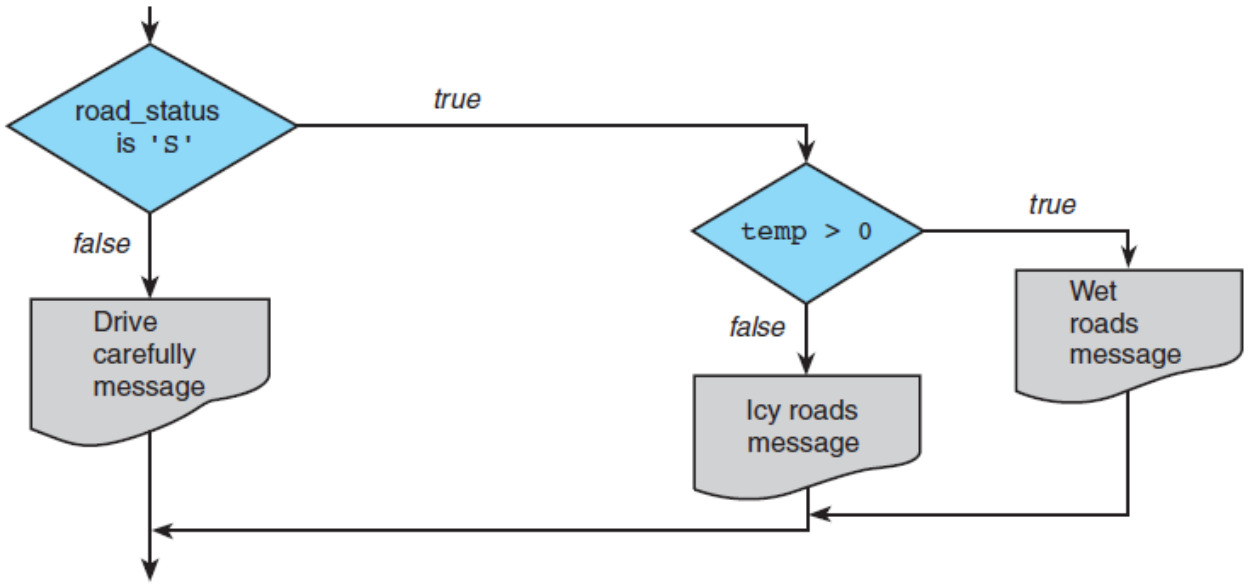
```
if (marital_status == 'S'  &&  gender == 'M'
    &&  age >= 18  &&  age <= 26)
         printf("All criteria are met.\n");
```



choose the correct message. The nested `if` statement below summarizes the decision process you should follow; the flowchart in Fig. 4.12 diagrams the process.

```
if (road_status == 'S')
      if (temp > 0) {
            printf("Wet roads ahead\n");
            printf("Stopping time doubled\n");
      } else {
            printf("Icy roads ahead\n");
            printf("Stopping time quadrupled\n");
      }
else
      printf("Drive carefully!\n");
```

Another way of writing the same if statement




```
if (road_status == 'D') {
      printf("Drive carefully!\n");
} else if (temp > 0) {
      printf("Wet roads ahead\n");
      printf("Stopping time doubled\n");
} else {
      printf("Icy roads ahead\n");
      printf("Stopping time quadrupled\n");
}
```

Whose “else”

```
/* incorrect interpretation of nested if */
if (road_status == 'S')
      if (temp > 0) {
            printf("Wet roads ahead\n");
            printf("Stopping time doubled\n");
      }
else
      printf("Drive carefully!\n");
```

```
/* correct interpretation of nested if */
if (road_status == 'S')
      if (temp > 0) {
            printf("Wet roads ahead\n");
            printf("Stopping time doubled\n");
      } else
            printf("Drive carefully!\n");
```

```
/* interpretation with braces around first true task */
if (road_status == 'S') {
      if (temp > 0) {
            printf("Wet roads ahead\n");
            printf("Stopping time doubled\n");
      }
} else
      printf("Drive carefully!\n");
```

17- What is the output of the following program segment?

```
int x = -1;
if (x++==0)  printf("%d\n",x);
else if(++x>1) printf("%d",x);
else printf("%d",x);
```

a) -1 b) 0 c) 1 d) 2 e) 3

---

18. For what exact range of values of variables a and b, does the following code segment display the value 0?

```
m= -1;
if (a>20)
  if (b<10)
    if (a>=30)
        m =4;
    else
        m=0;
  else
      m=1;
else
   m=2;
printf("%d",m);
```

a) $a > 20$, $b \geq 10$

b) $20 \leq a \leq 30$, $b \leq 10$

c) $20 < a < 30$, $b < 10$

d) $a \geq 30$, $b < 10$

e) $20 < a < 30$, $b \geq 10$

19- Assuming that x,y and flag are integers, what is the value printed by the following if statements?

```
if(x>y)
 if(x>z) printf("%d", x);
 else
  if(z>y) printf("%d",z);
  else printf("%d", y);
else
 if(y>z) printf("%d",y);
 else printf("%d",z);
```

a) minimum b) maximum c) median
d) last e) indeterminate

20- What is the output of the following program segment?

```
int x=6, y=3, A=3, B=5, C=7;
if (x <A && y >B)
 if (y >0)
    printf("A");
 else printf("B");
else if (y>C || x >0)
    printf("C");
```

a) A b) B c) C d) AC e) no output

27) What will be the output of the program?

```
#include<stdio.h>
void main( ){
   int a = 4;
   if(a == 4)
      printf("a1");
   else
      printf("a2");
         printf("a3");
   printf("a4"); }
```

Scope of else without {}

a) a4 **b)** a1a4 **c)** a2a3a4 **d)** a1a3a4 **e)** a2a3

---

28) What will be the output of the program?

```
#include<stdio.h>
void main()
{
   int a = 9, b = 3;
   if( !a <= 4 )
      b = 5;
      a = 1;
   printf("a=%d b=%d\n", a, b); }
```

Scope of if without {}

a) a = 9, b = 3

b) a = 4, b = 3

c) a = 1, b = 5

d) a = 9, b = 5

e) a = 1, b = 3

32) What will be the output of the program?

```
#include<stdio.h>
void main( ){
  int m=8;
  float n=8.6;
  if (m > n)
    { }
  else {
    m = n * 2;
    n = n / 2; }
   printf(" %d %f ", m, n);
 }
```

a) 17 4.300000
b) 17 4.000000
c) 16 4.300000
d) 16 4.000000
e) Compile error

---

31) What will be the output of the program?

```
#include<stdio.h>
void main( ){
   int z=9;
   z=z-4;
   if( z<9 || ++z>4 ) z=z+2;
   printf(" %d ", z);
   }
```

a) 5 **b)** 6 **c)** 7 **d)** 8 **e)** 9

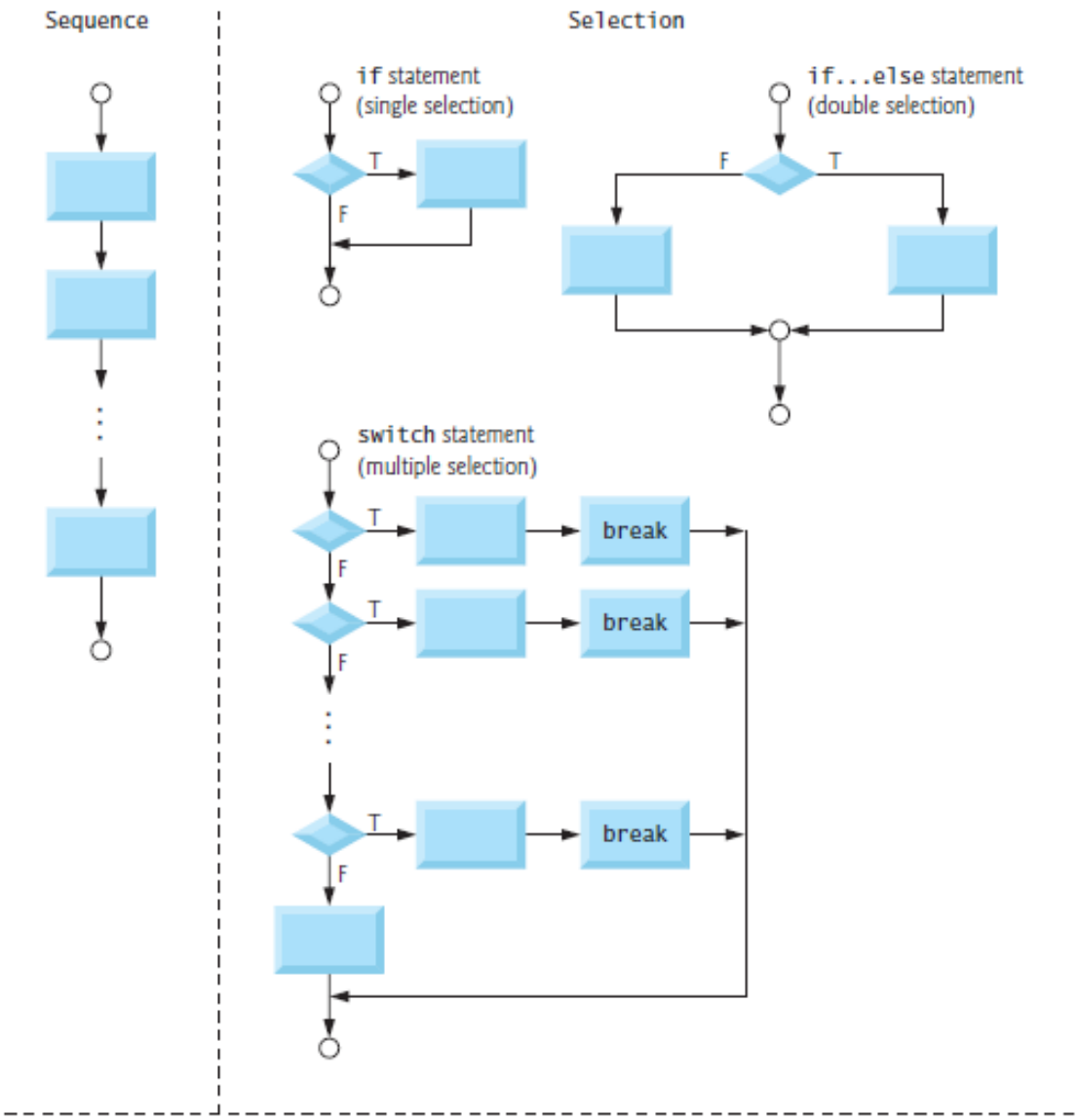
```
if (0.11)  // nonzero value
            printf(“ It is true when (0.11)");
else
             printf(“ It is false when (0.11)");
```

This code prints : It is true when (0.11)

char next_ch='A';

```
switch (next_ch) {
case 'A':
case 'a':
     printf("Excellent");
     break;

case 'B':
case 'b':
     printf("Good");
     break;

case 'C':
case 'c':
     printf("O.K.");
     break;

case 'D':
case 'd':
case 'F':
case 'f':
     printf("Poor, student is ");
     printf("on probation");
     break;

default:
     printf("Invalid letter grade");
}
```

Displays one of five messages based on the value of `next_ch` (type `char`). If `next_ch` is `'D'`, `'d'`, or `'F'`, `'f'`, the student is put on probation. If `next_ch` is not listed in the `case` labels, displays an error message.


Sequence
Selection
if statement
(single selection)
T
F
if...else statement
(double selection)
F
T
switch statement
(multiple selection)
T
F
break
T
F
break
T
F
break

```c
switch ( value % 2 ) {
   case 0:
     printf( "Even integer\n" );
   case 1:
     printf( "Odd integer\n" );
}
```

please look at the the sample codes for **switch** statement that are given on my web site.

Homework

Correct the following program code segment with <u>new</u> **else if** statements :

```
/* incorrect perception of noise loudness */

if (noise_db <= 110)
      printf("%d-decibel noise is very annoying.\n", noise_db);
else if (noise_db <= 90)

      printf("%d-decibel noise is annoying.\n",
              noise_db);
else if (noise_db <= 70)
      printf("%d-decibel noise is intrusive.\n",
              noise_db);
else if (noise_db <= 50)
      printf("%d-decibel noise is quiet.\n",
              noise_db);
else
      printf("%d-decibel noise is uncomfortable.\n", noise_db);
```

Upload the .c file to LMS.