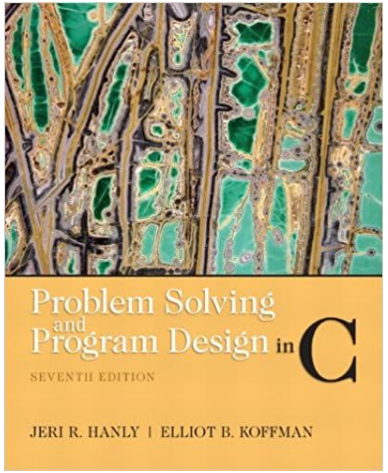
جامعة بيرزيت
BIRZEIT UNIVERSITY

Faculty of Engineering and Technology
Department of Computer Science

Introduction to Computers and Programming (Comp 133)


Problem Solving and Program Design in C
SEVENTH EDITION
JERI R. HANLY | ELLIOT B. KOFFMAN

References :
Book : Problem Solving and Program Design in C (7th Edition)
Slides : Dr. Radi Jarrar , Dr. Abdallah Karakra , Dr. Majdi Mafarja.

Selection Structures: If and Switch

Chapter 4

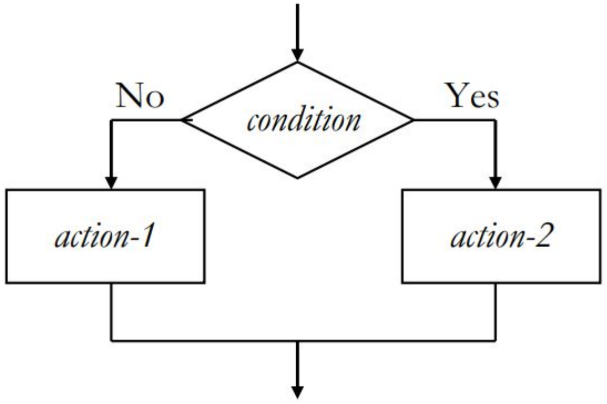
Control structures

- **Control structure** a combination of individual instructions into a single logical unit with one entry point and one exit point.
- **Compound statement** a group of statements bracketed by { and } that are executed sequentially.
- **Selection control structure** a control structure that chooses among alternative program statements.
- **Condition** an expression that is either false (represented by 0) or true (usually represented by 1)

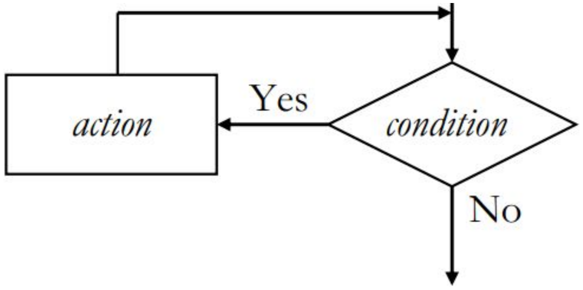
Control structures

Types of control structures

- Sequence
  - Programs executed statements sequentially
- Selection
  - If
  - if...else
  - switch
- Repetition
  - While
  - do...while
  - for



Selection structure



Repetition structure

Chapter 4

- Selection Structures

Selection Structures

Relational and Equality Operators

- variable relational-operator variable (x>y)
- variable relational-operator constant (x>10)
- variable equality-operator variable (x==y)
- variable equality-operator constant (x!=5)

Selection Structures

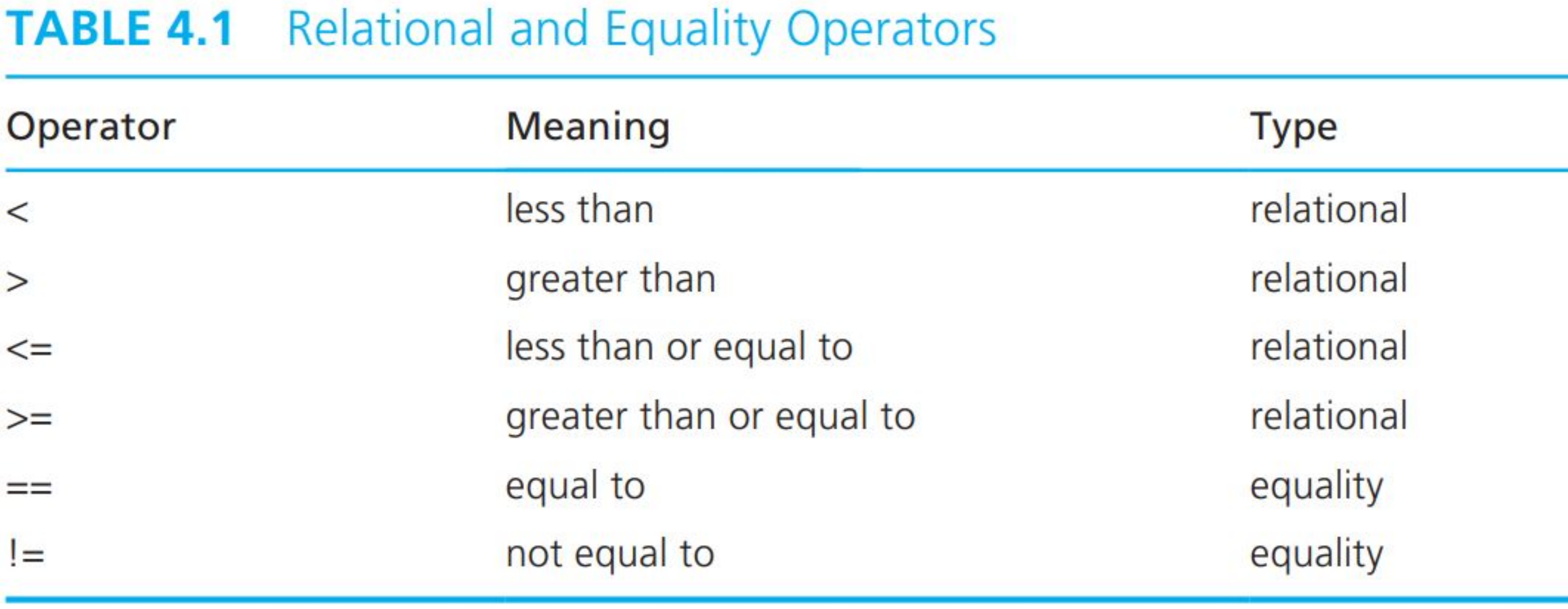


TABLE 4.1 Relational and Equality Operators

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

Relational and Equality Operators example

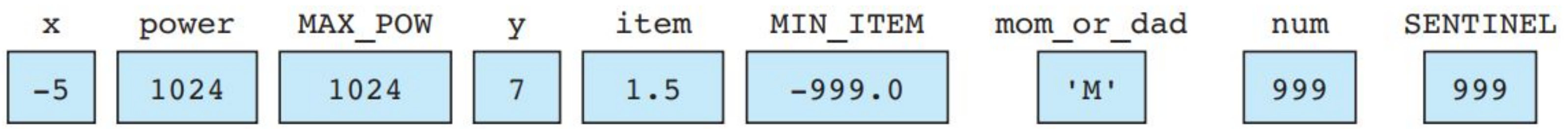


TABLE 4.2 Sample Conditions

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)

Logical Operators

- There are three logical operators
  - **&&** **(and)**
  - **||** **(or)**
  - **!** **(not)**
- We can form more complicated conditions or logical expressions.
- **logical expression** an expression that uses one or more of the logical operators && (and), || (or), ! (not).

Logical Operators

X	**Y**	**X&&Y**	**X||Y**	**!X**
T	T	T	T	F
T	F	F	T	F
F	T	F	T	T
F	F	F	F	T

Operator Precedence

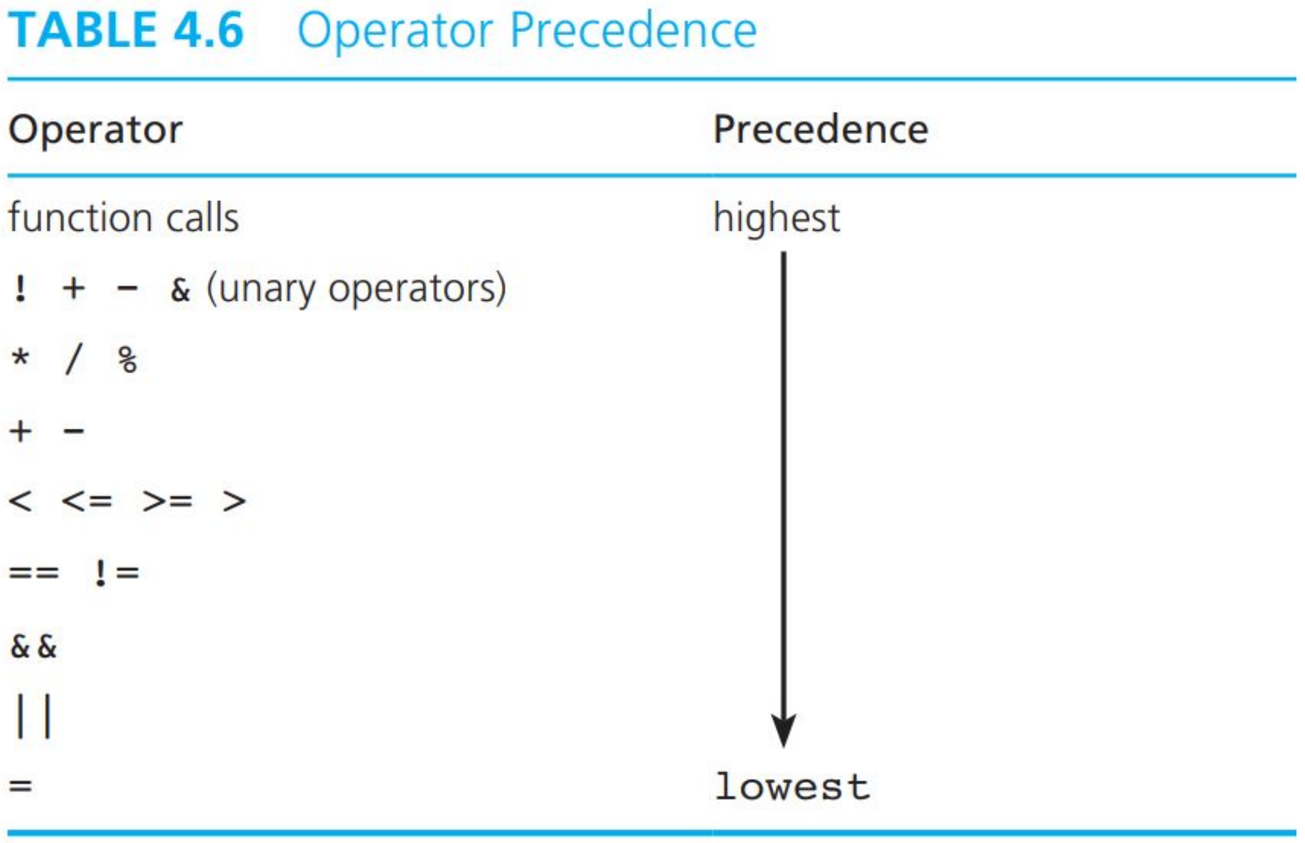
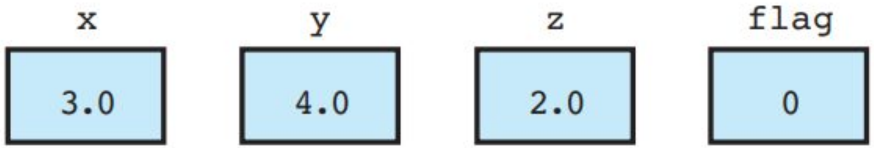


TABLE 4.6 Operator Precedence

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

Operator Precedence example



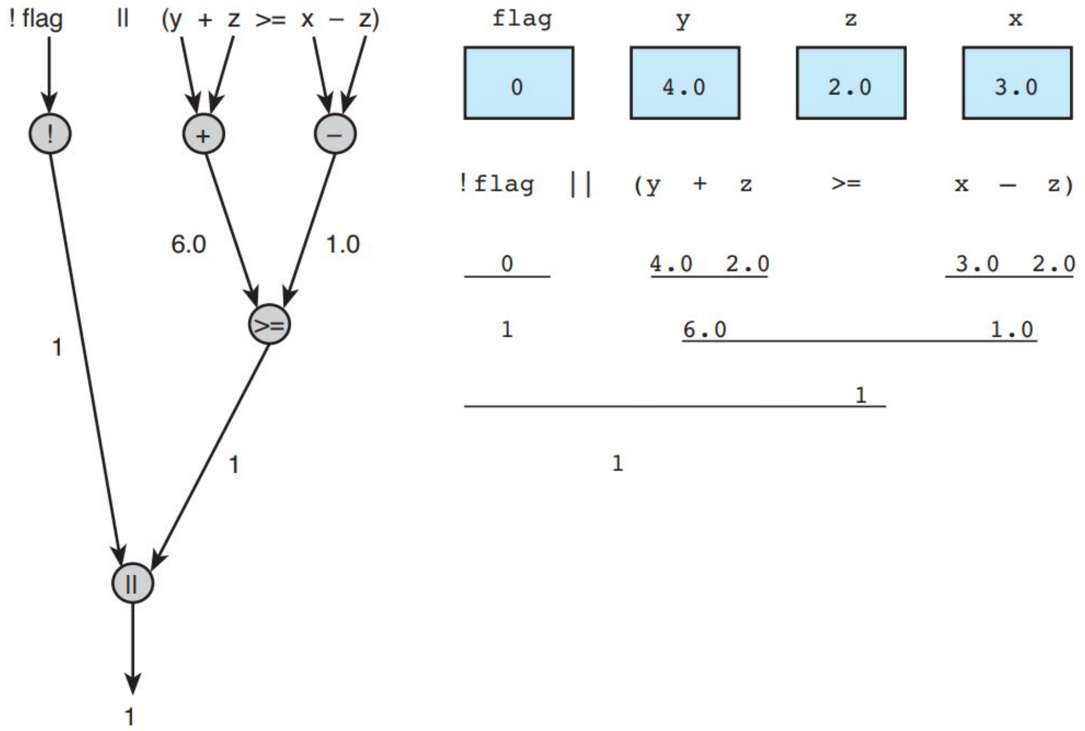
```
1.  !flag                                /* !0 is 1 (true)             */
2.  x + y / z  <=  3.5                   /* 5.0 <= 3.5 is 0 (false)    */
3.  !flag || (y + z  >=  x - z)          /* 1 || 1 is 1 (true)         */
4.  !(flag || (y + z  >=  x - z))        /* !(0 || 1) is 0 (false)     */
```

Figure 4.1 shows the evaluation tree and step-by-step evaluation for expression 3.

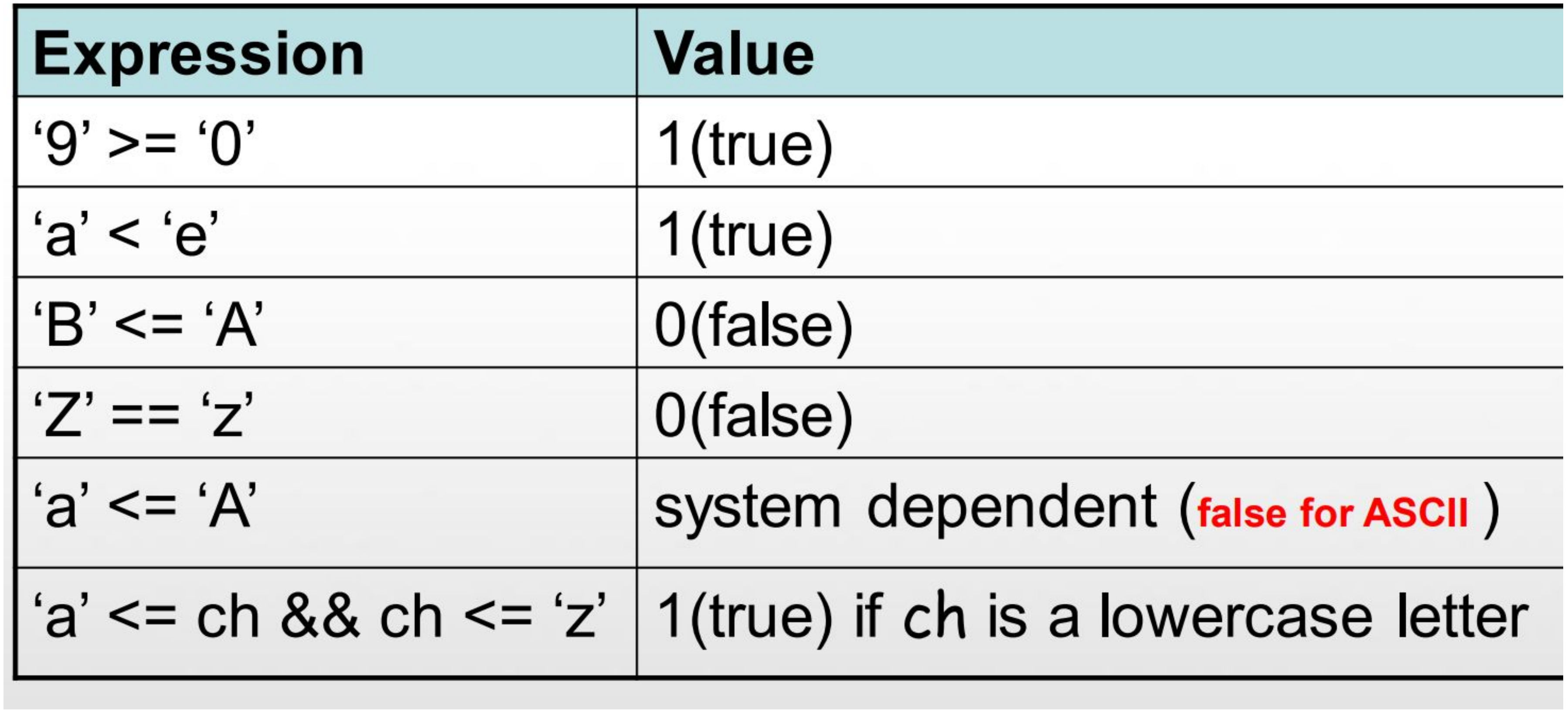
Operator Precedence example

Evaluation Tree and Step-by-Step Evaluation for

- !flag || (y + z >= x − z)

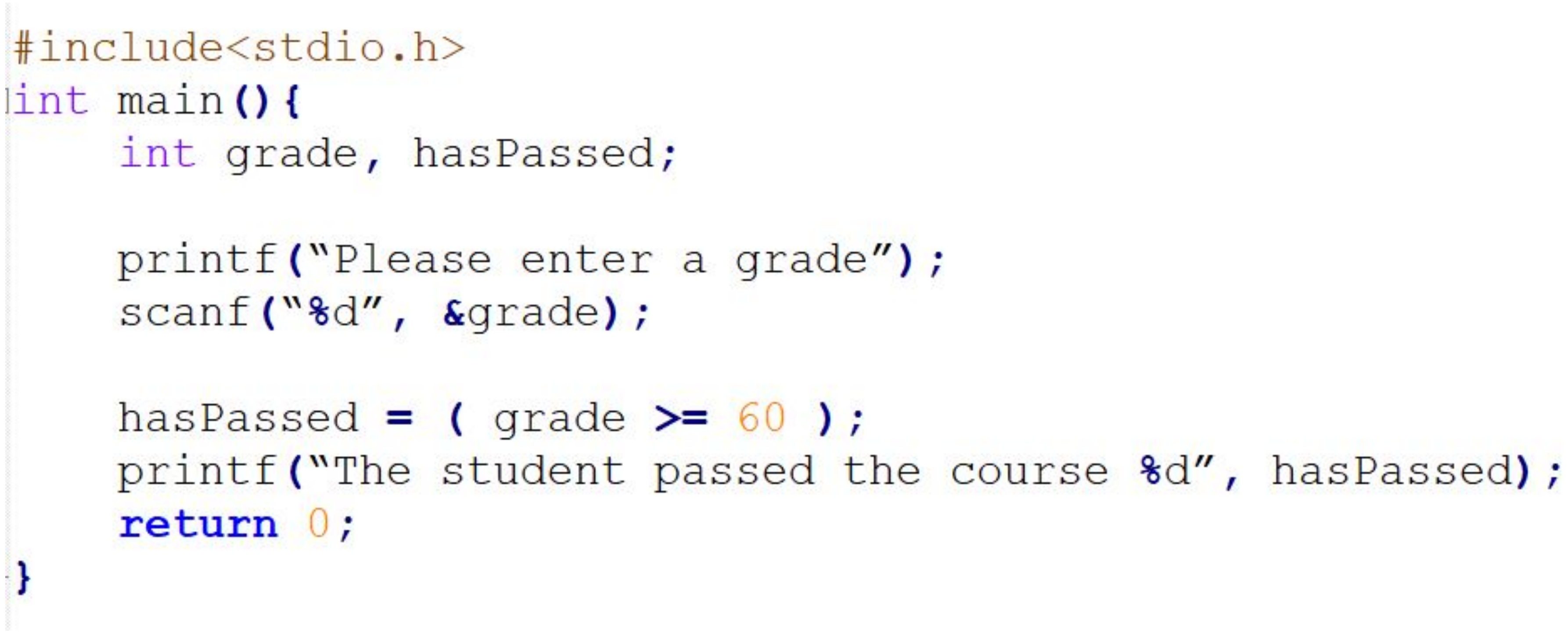


Comparing Characters



Expression	Value
‘9’ >= ‘0’	1(true)
‘a’ < ‘e’	1(true)
‘B’ <= ‘A’	0(false)
‘Z’ == ‘z’	0(false)
‘a’ <= ‘A’	system dependent (false for ASCII)
‘a’ <= ch && ch <= ‘z’	1(true) if ch is a lowercase letter

Logical Assignment



```
#include<stdio.h>
int main(){
    int grade, hasPassed;

    printf(“Please enter a grade”);
    scanf(“%d”, &grade);

    hasPassed = ( grade >= 60 );
    printf(“The student passed the course %d”, hasPassed);
    return 0;
}
```

Chapter 4

- Selection Structures
    - If Statement

If statement

- If statement with **one alternative**

```
if(expression)    //  if(x==10)
 Statement;
```

- If statement with **two alternatives**

```
if(expression)   //  if(age>56)
 Statement;
else
 Statement;
```

If statement

- **If with single-statement**

```
if (x != 0.0)
product = product * x;  // This statement execute if true only
printf("%f",product );  // every time execute not related to if
```

- **If with compound-statement**

```
if(x!=0.0)
{
 product = product * x;
printf("%f",product );
}
```

All statements between braces execute.

```
if ( condition )
{
 true task
}
else
{
 false task
}
```

If statement

A hand trace, or desk check

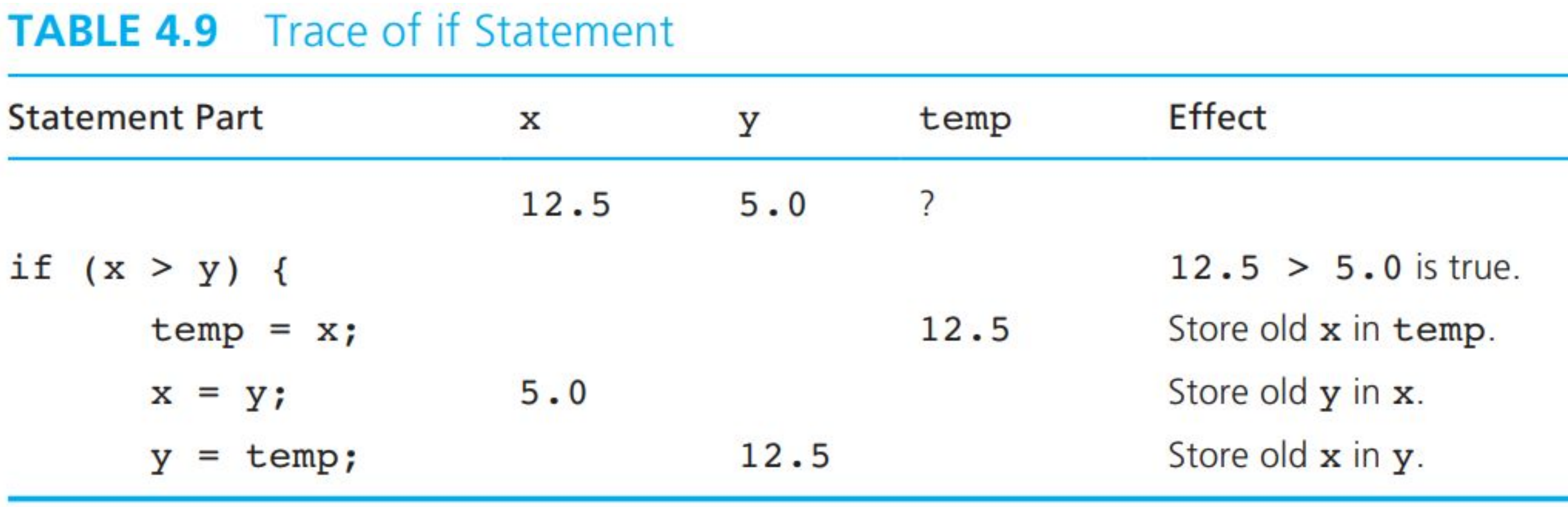
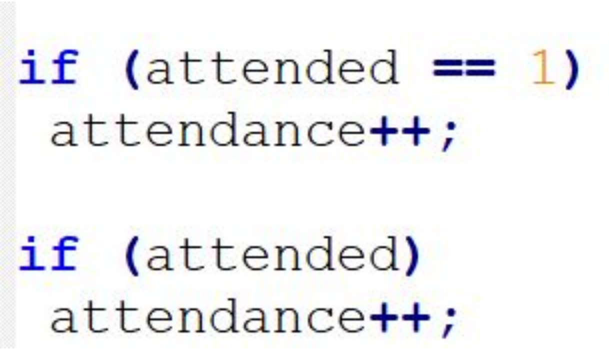


TABLE 4.9 Trace of if Statement

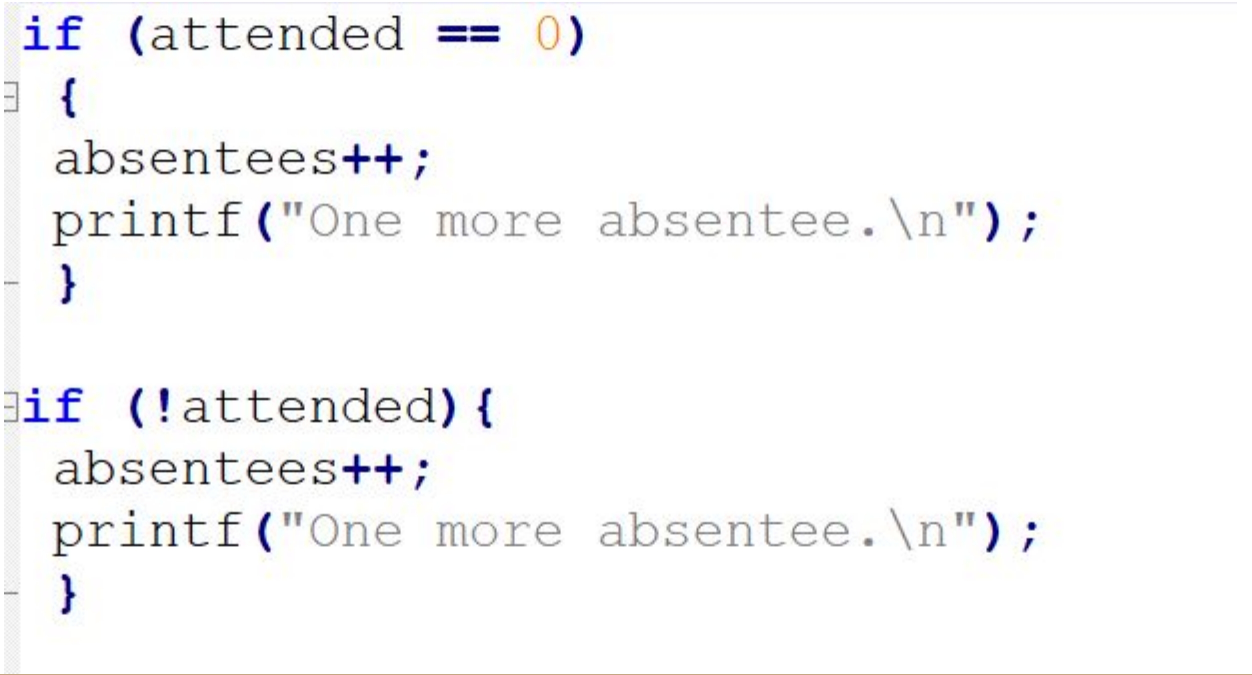
Statement Part	x	y	temp	Effect
	12.5	5.0	?	
`if (x > y) {`				`12.5 > 5.0` is true.
`temp = x;`			12.5	Store old `x` in `temp`.
`x = y;`	5.0			Store old `y` in `x`.
`y = temp;`		12.5		Store old `x` in `y`.

If statement

Flags : is a variable whose contents an integer variable with zero for "false" and non-zero for "true"



```
if (attended == 1)
 attendance++;

if (attended)
 attendance++;
```



```
if (attended == 0)
 {
 absentees++;
 printf("One more absentee.\n");
 }

if (!attended){
 absentees++;
 printf("One more absentee.\n");
 }
```

If statement

Nested if Statements 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;
```

Multiple-Alternative Decisions

If statement

Nested if Statements with More Than One Variable

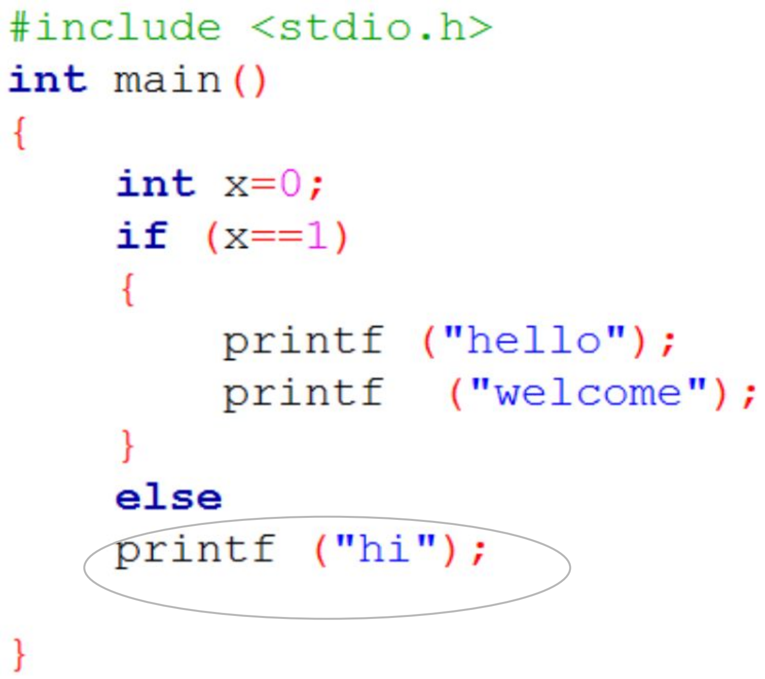
```
/* Print a message if all criteria are met. */
if (marital_status == 'S')
 if (gender == 'M')
  if (age >= 18 && age <= 26)
   printf("All criteria are met.\n");
```

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

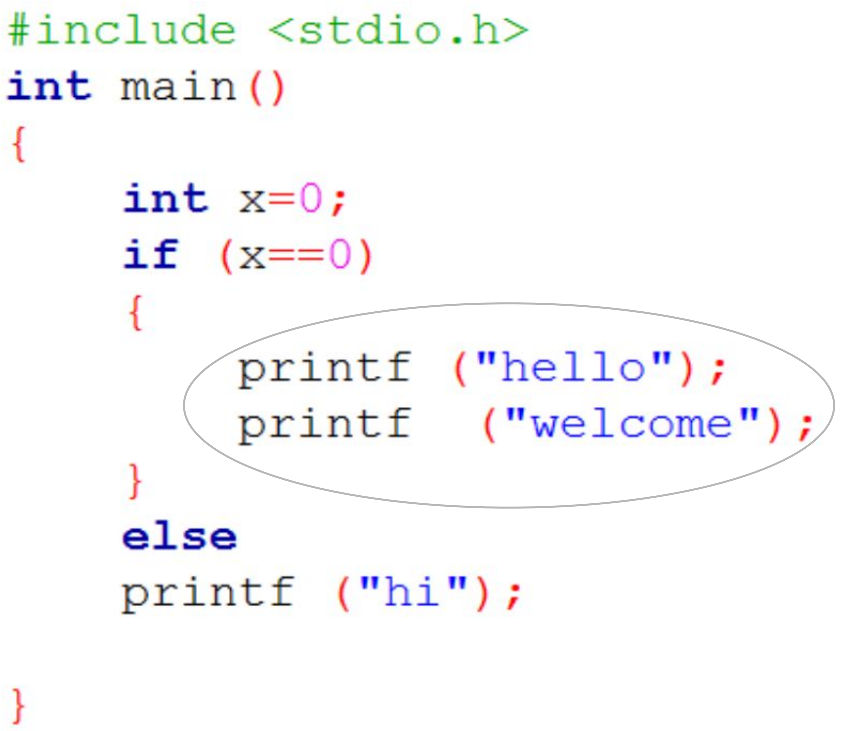
```
if (marital_status == 'S' && gender == 'M' && age >= 18 && age <= 26)
 printf("All criteria are met.\n");
```

If statement

What is the output of the following program?



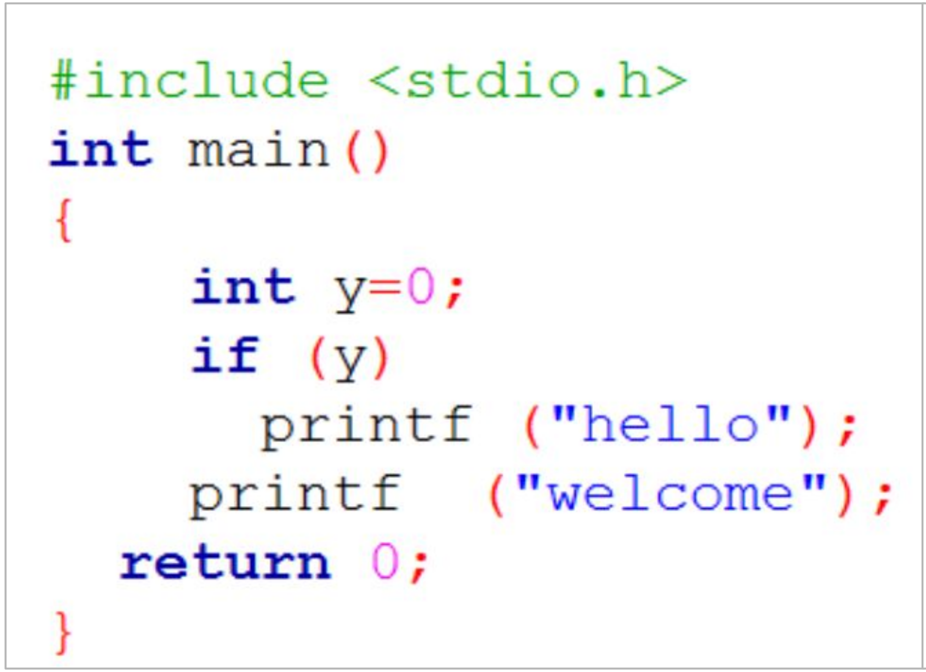
```c
#include <stdio.h>
int main()
{
     int x=0;
     if (x==1)
     {
         printf ("hello");
         printf  ("welcome");
     }
     else
     printf ("hi");

}
```



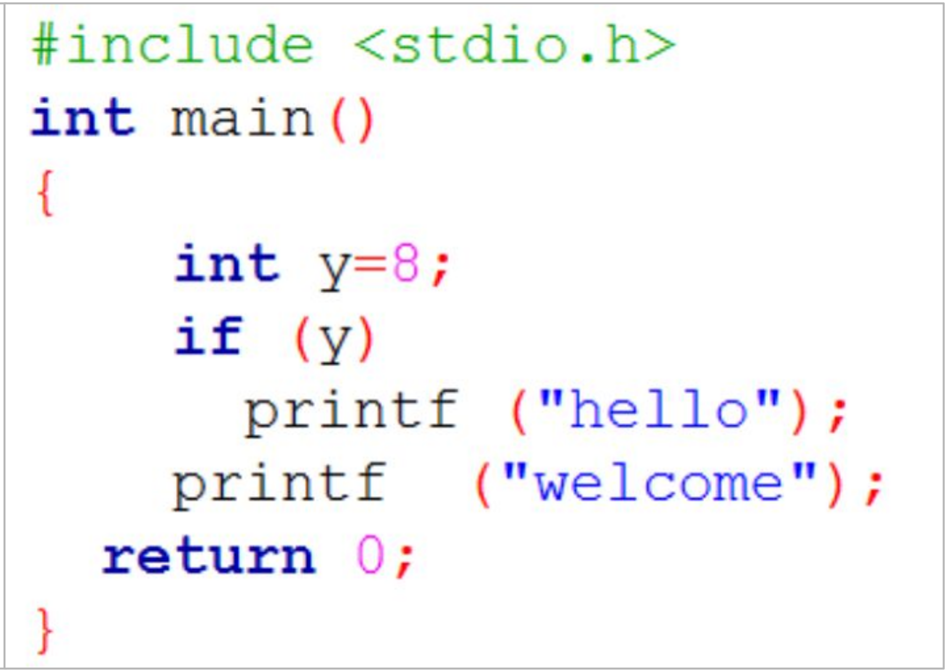
```c
#include <stdio.h>
int main()
{
     int x=0;
     if (x==0)
     {
         printf ("hello");
         printf  ("welcome");
     }
     else
     printf ("hi");

}
```

If statement

What is the output of the following program?



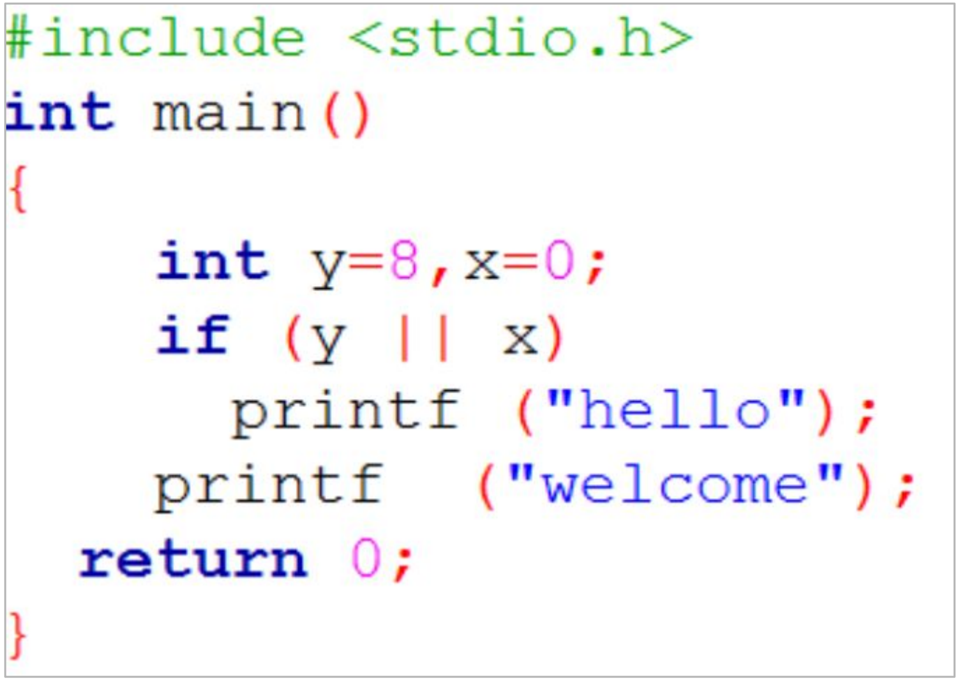
```c
#include <stdio.h>
int main()
{
      int y=0;
      if (y)
         printf ("hello");
      printf  ("welcome");
   return 0;
}
```



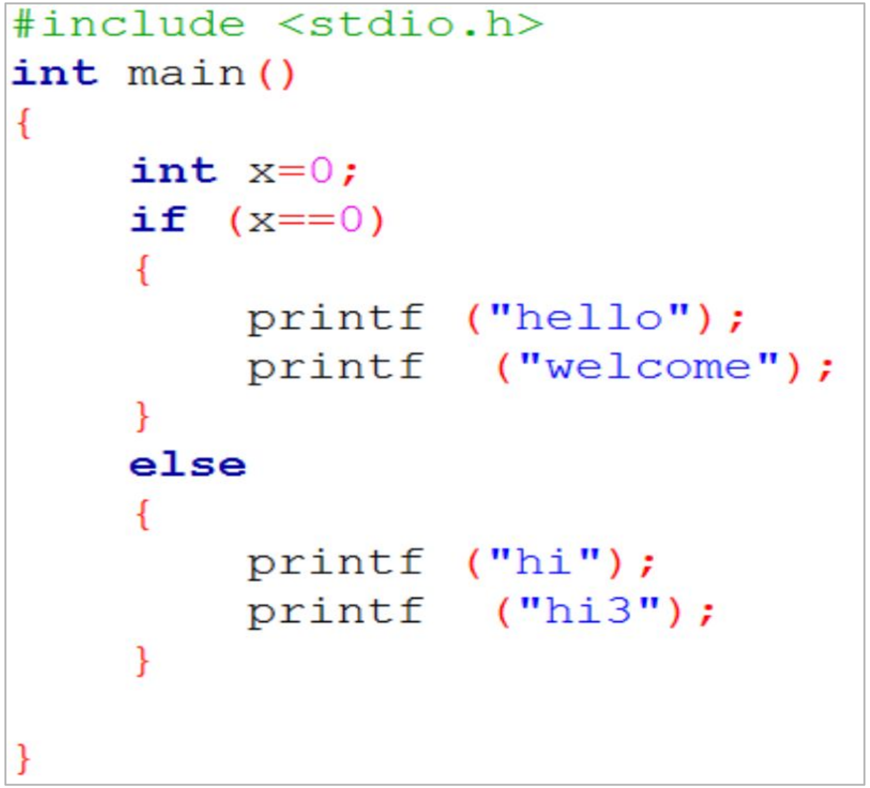
```c
#include <stdio.h>
int main()
{
      int y=8;
      if (y)
         printf ("hello");
      printf  ("welcome");
   return 0;
}
```

If statement

What is the output of the following program?



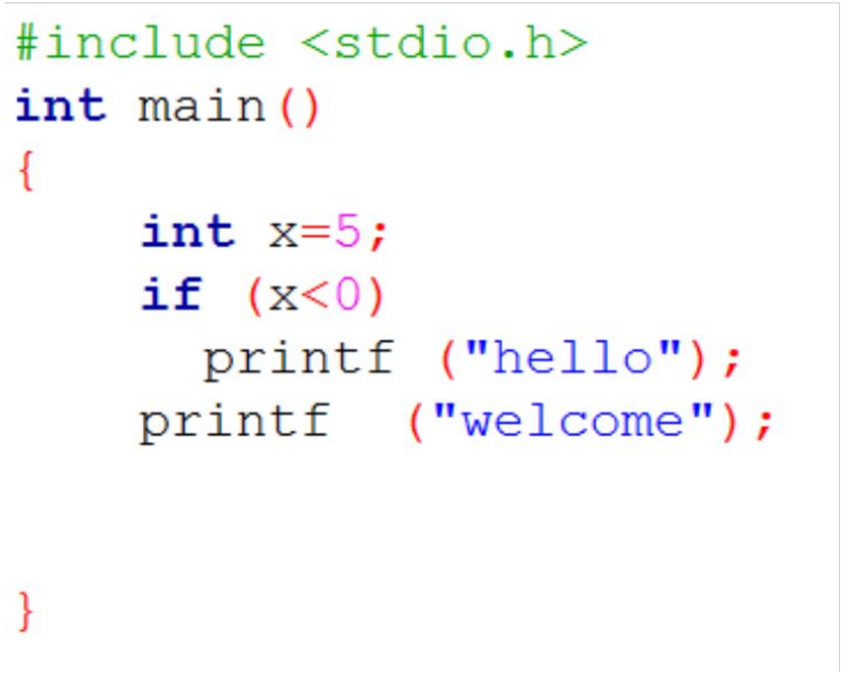
```
#include <stdio.h>
int main()
{
        int y=8,x=0;
        if (y || x)
          printf ("hello");
        printf  ("welcome");
    return 0;
}
```



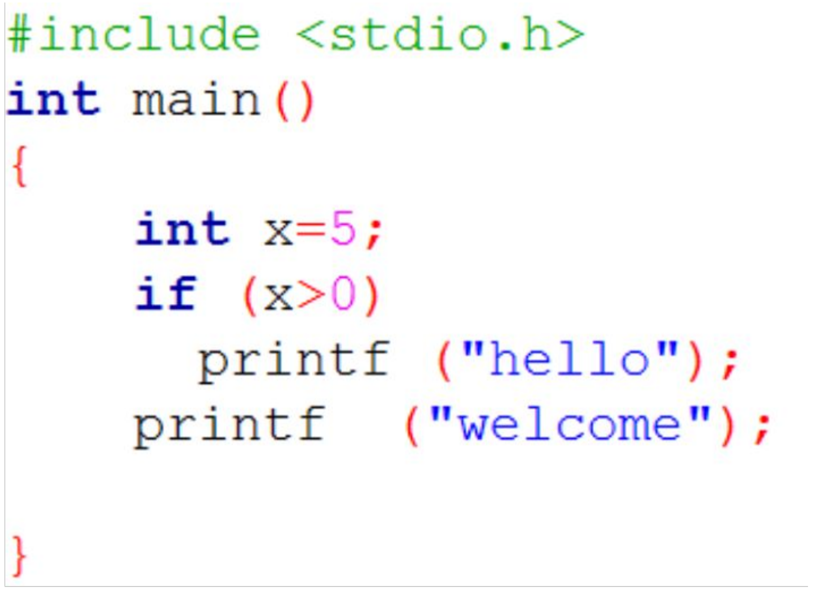
```
#include <stdio.h>
int main()
{
        int x=0;
        if (x==0)
        {
              printf ("hello");
              printf  ("welcome");
        }
        else
        {
              printf ("hi");
              printf  ("hi3");
        }

}
```

If statement

What is the output of the following program?



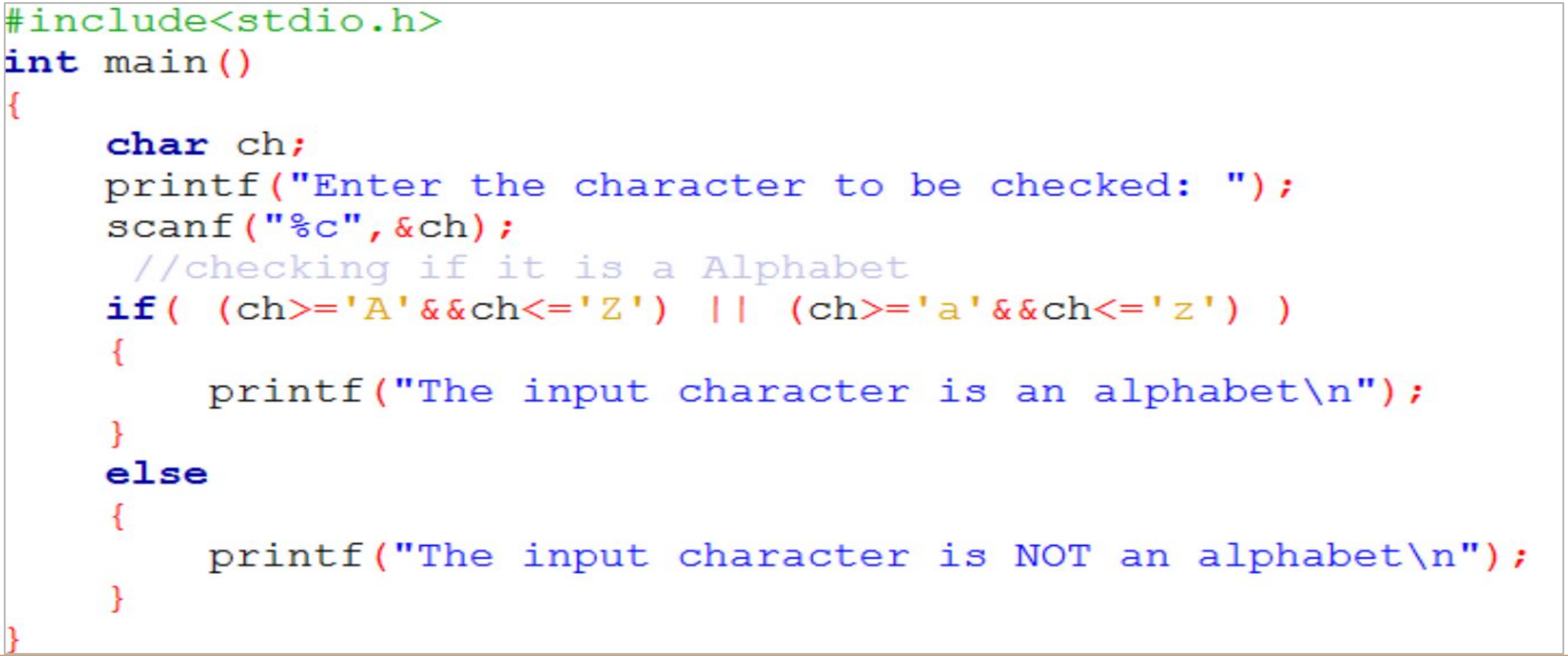
```c
#include <stdio.h>
int main()
{
    int x=5;
    if (x<0)
      printf ("hello");
    printf  ("welcome");


}
```



```c
#include <stdio.h>
int main()
{
    int x=5;
    if (x>0)
      printf ("hello");
    printf  ("welcome");

}
```

If statement Example

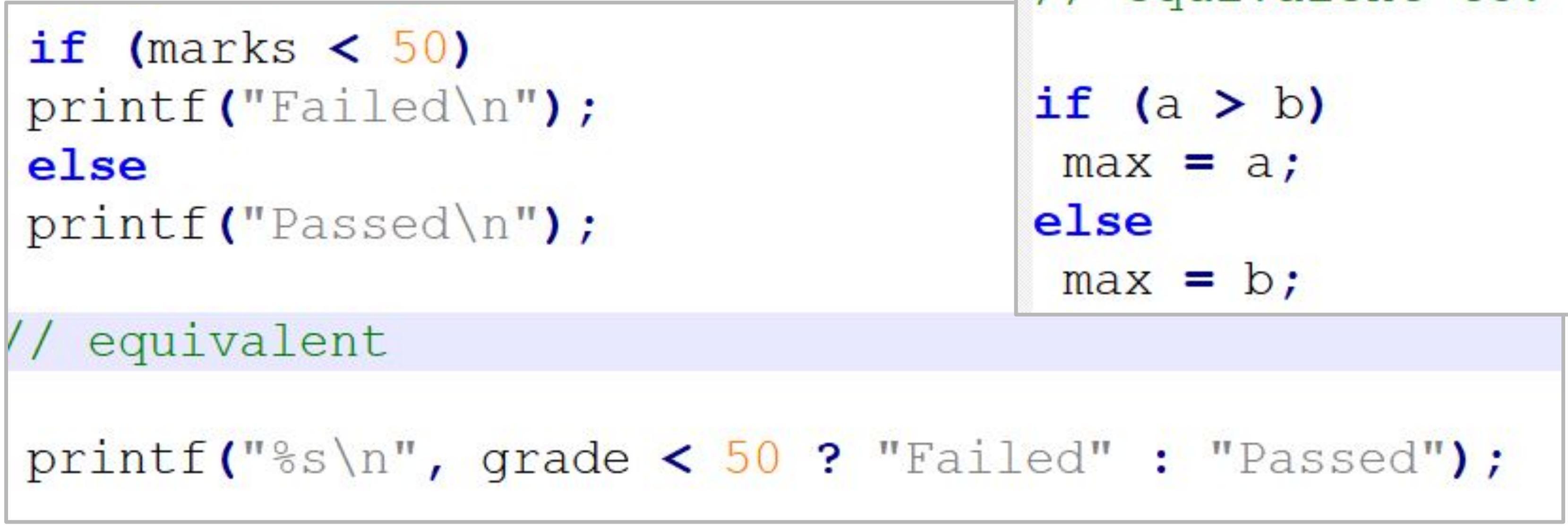
Write a C program which takes a character as input from the user. Check whether the character is an alphabet or not.



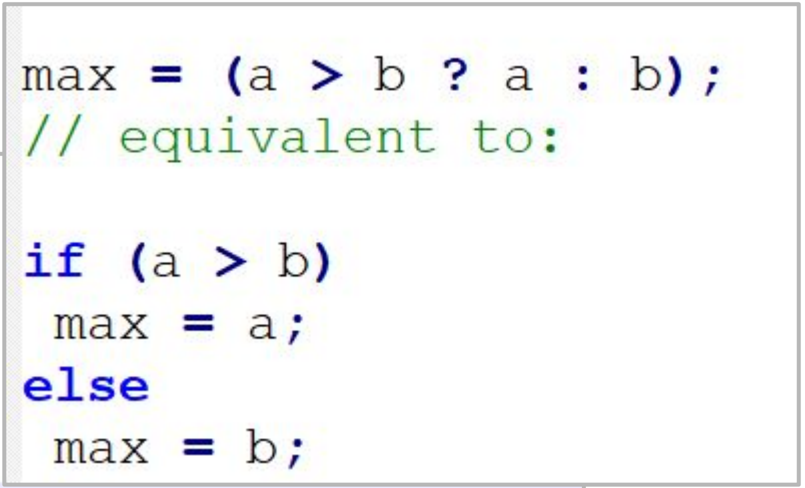
```c
#include<stdio.h>
int main()
{
    char ch;
    printf("Enter the character to be checked: ");
    scanf("%c",&ch);
     //checking if it is a Alphabet
    if( (ch>='A'&&ch<='Z') || (ch>='a'&&ch<='z') )
    {
        printf("The input character is an alphabet\n");
    }
    else
    {
        printf("The input character is NOT an alphabet\n");
    }
}
```

Conditional Operator (?:)

- **Syntax** : condition ? expr1 : expr2



```c
if (marks < 50)
printf("Failed\n");
else
printf("Passed\n");

// equivalent

printf("%s\n", grade < 50 ? "Failed" : "Passed");
```



```c
max = (a > b ? a : b);
// equivalent to:

if (a > b)
 max = a;
else
 max = b;
```

Chapter 4

- Selection Structures
    - Switch Statement

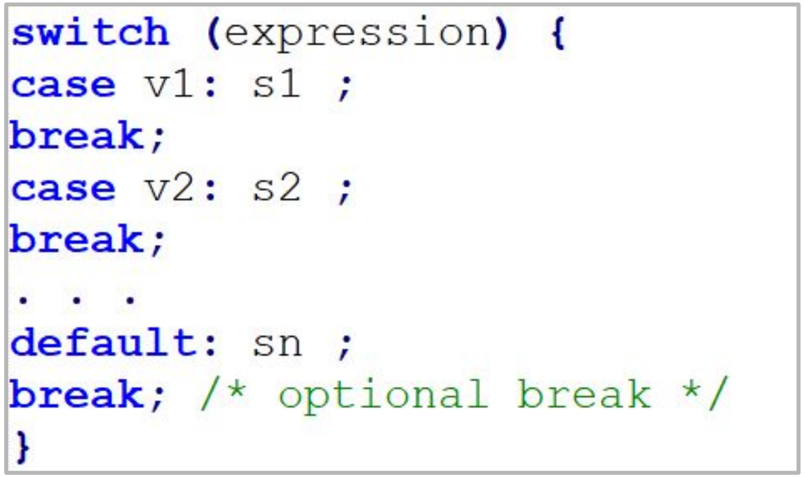
If-else is more efficient than If ?

```
if (day == 0 )
 printf ("Sunday") ;
if (day == 1 )
 printf ("Monday") ;
if (day == 2)
 printf ("Tuesday") ;
if (day == 3)
 printf ("Wednesday") ;
if (day == 4)
 printf ("Thursday") ;
if (day == 5)
 printf ("Friday") ;
if (day == 6)
 printf ("Saturday") ;
if ((day < 0) || (day > 6))
 printf("Error - invalid day.\n");
```

```
if (day == 0 ) {
 printf ("Sunday") ;
} else if (day == 1 ) {
 printf ("Monday") ;
} else if (day == 2) {
 printf ("Tuesday") ;
} else if (day == 3) {
 printf ("Wednesday") ;
} else if (day == 4) {
 printf ("Thursday") ;
} else if (day == 5) {
 printf ("Friday") ;
} else if (day = 6) {
 printf ("Saturday") ;
} else {
 printf ("Error - invalid day.\n”) ;
}
```

Switch Statement

- The switch is a multi-selection statement that could be used instead of **‘if-else’** statement.
- The switch statement selection is based on the value of a single variable or of a simple expression.
- The value of the expression should be of type **int** or **char** ONLY.



```
switch (expression) {
case v1: s1 ;
break;
case v2: s2 ;
break;
. . .
default: sn ;
break; /* optional break */
}
```

Switch Statement

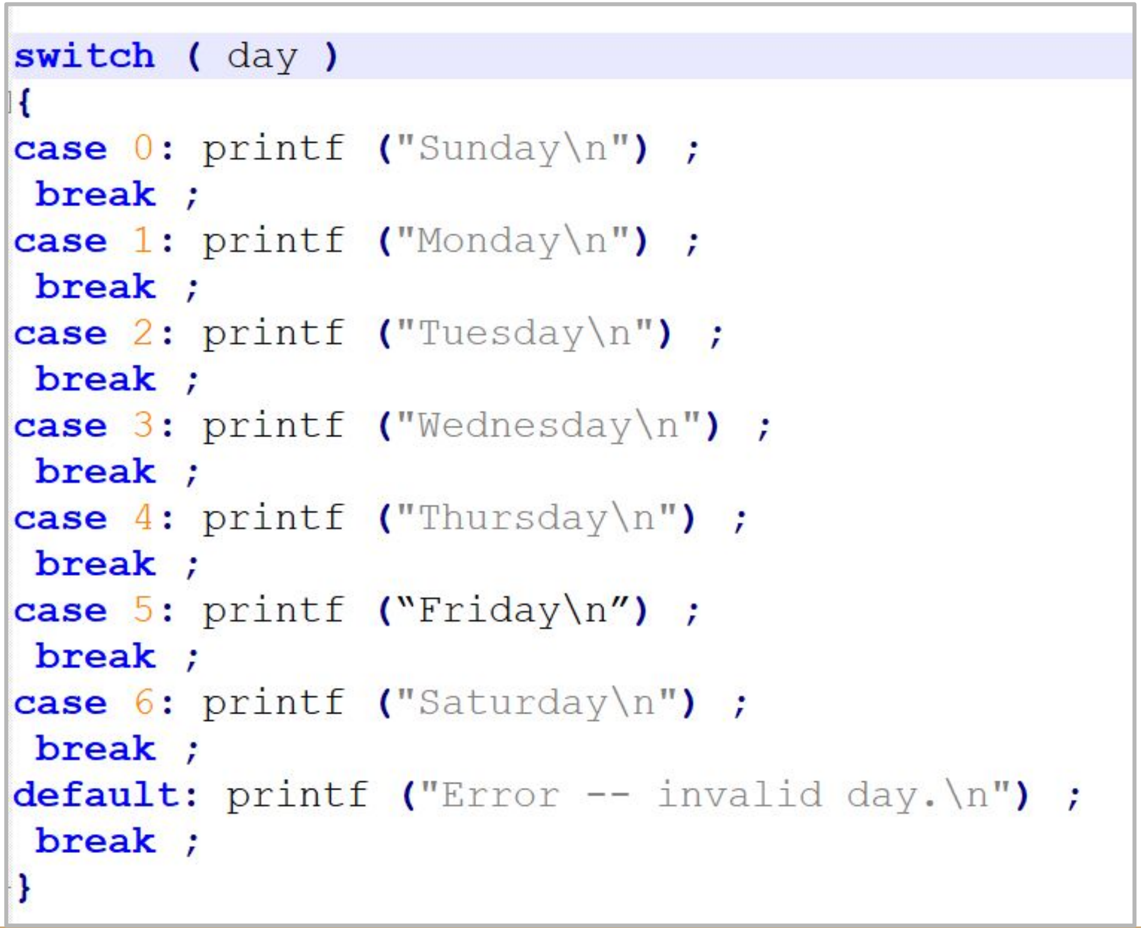
- A **break** should almost be after each **case** in the switch.
- The break causes program to jump to the next line after the end brace of switch.
- **Without the break**, the code flows into the next case. This conflicts with the goal of switch structure.
- **default** is optional , but always consider using it.
- Switch statement may be easier to read.
- Switch is easier to add new cases to a switch statement than to a nested if-else structure.

Switch Statement

What is the output when **day=3** ?

Output :

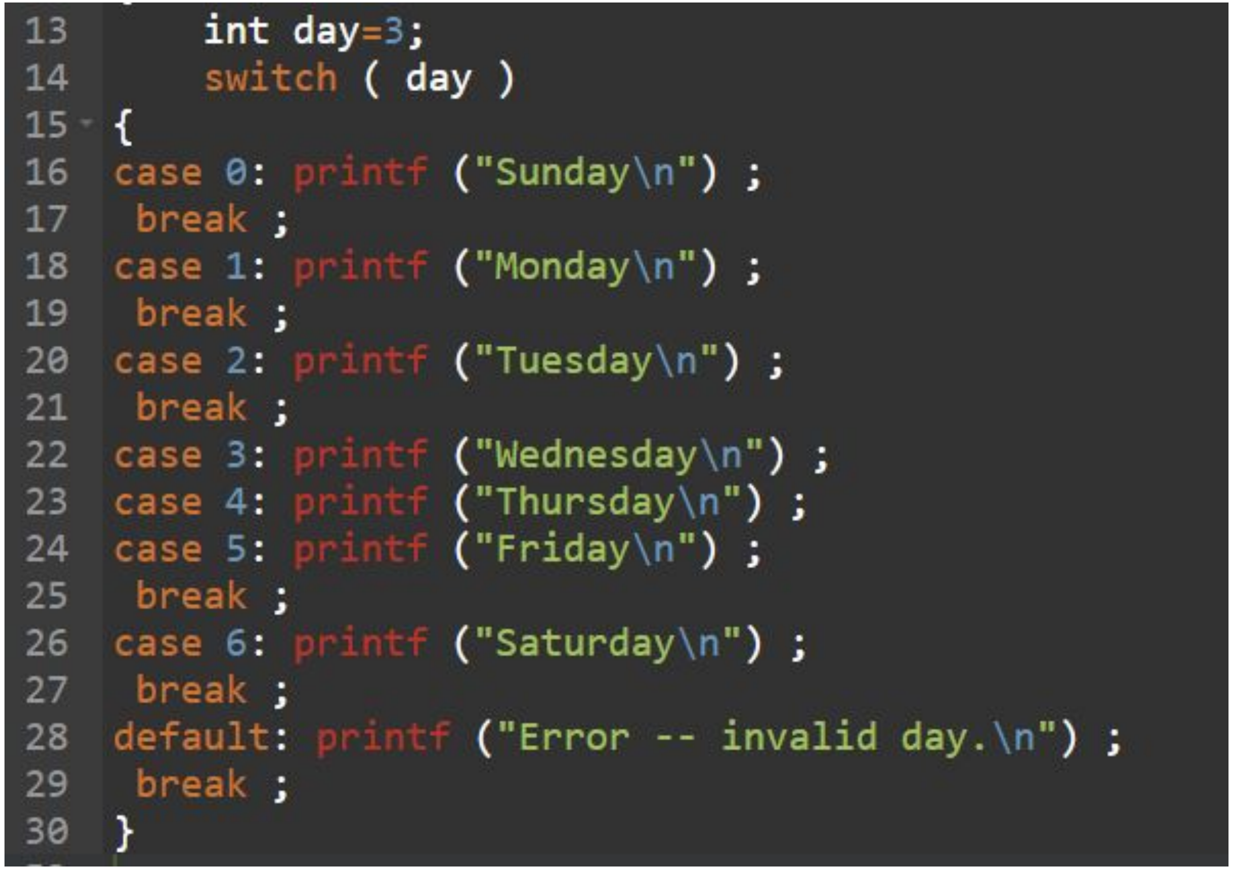
Wednesday



```
switch ( day )
{
case 0: printf ("Sunday\n") ;
 break ;
case 1: printf ("Monday\n") ;
 break ;
case 2: printf ("Tuesday\n") ;
 break ;
case 3: printf ("Wednesday\n") ;
 break ;
case 4: printf ("Thursday\n") ;
 break ;
case 5: printf (“Friday\n”) ;
 break ;
case 6: printf ("Saturday\n") ;
 break ;
default: printf ("Error -- invalid day.\n") ;
 break ;
}
```

Switch Statement

What is the output when **day=3** ?



```
    int day=3;
    switch ( day )
{
case 0: printf ("Sunday\n") ;
 break ;
case 1: printf ("Monday\n") ;
 break ;
case 2: printf ("Tuesday\n") ;
 break ;
case 3: printf ("Wednesday\n") ;
case 4: printf ("Thursday\n") ;
case 5: printf ("Friday\n") ;
 break ;
case 6: printf ("Saturday\n") ;
 break ;
default: printf ("Error -- invalid day.\n") ;
 break ;
}
```

Output :

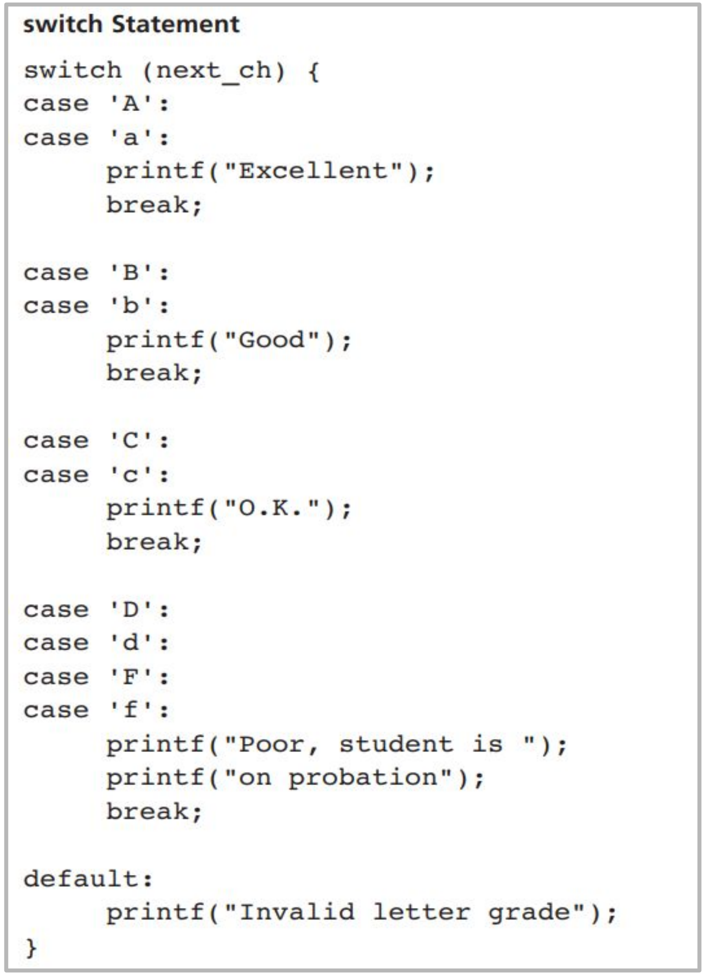
Wednesday

Thursday

Friday

Switch Example

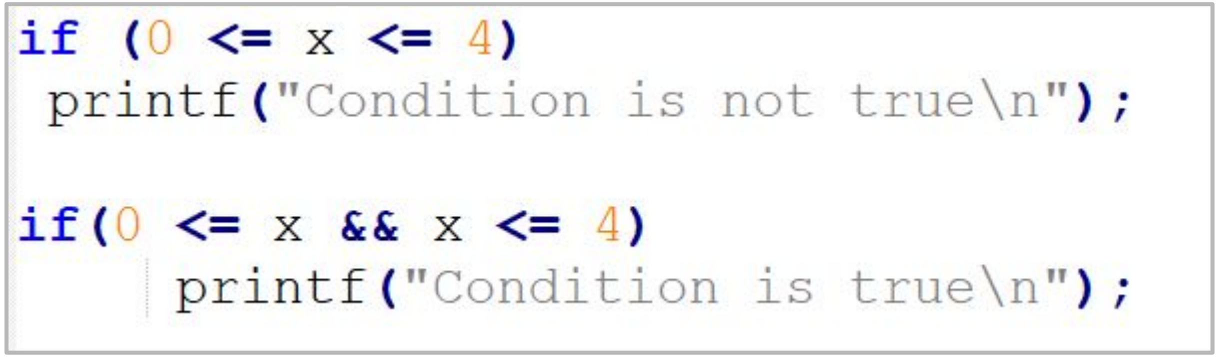
Displays one of five messages based on the value of next_ch (type char). If next_ch is **'D', 'd', or 'F', 'f'**, the student is put on probation. If next_ch is not listed in the case labels, displays an error message.



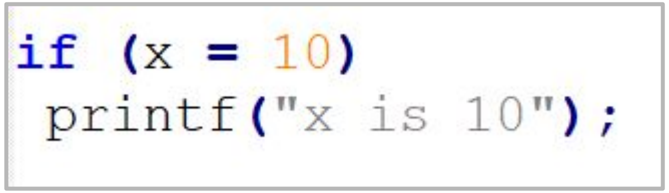
switch Statement

```
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");
}
```

Common Programming Errors

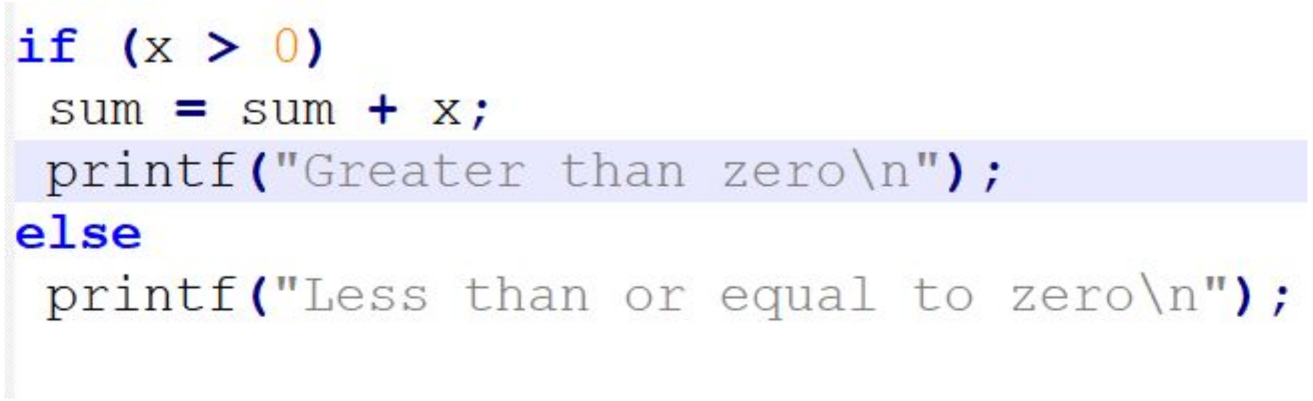


```
if (0 <= x <= 4)
 printf("Condition is not true\n");

if(0 <= x && x <= 4)
     printf("Condition is true\n");
```



```
if (x = 10)
 printf("x is 10");
```

Always prints x is 10 , regardless of the value of x .



```
if (x > 0)
 sum = sum + x;
 printf("Greater than zero\n");
else
 printf("Less than or equal to zero\n");
```

Thank You.