CONDITIONALS AND BRANCHING

CS10003 PROGRAMMING AND DATA STRUCTURES

Conditionals

Branching / Conditionals: What do they do?

- Allow different sets of instructions to be executed depending on the outcome of a logical condition / test

- Whether the condition is TRUE (non-zero) or FALSE (zero)

3

How do we specify the condition?

Using relational operators.

- **Four relation operators:** **<, <=, >, >=**
- **Two equality operations:** **==, !=**

Using logical operators / connectives.

- **Two logical connectives:** **&&, ||**
- **Unary negation operator:** **!**

EXAMPLES

(count <= 100)

((math+phys+chem) / 3 >= 60)

((sex == 'M') && (age >= 21))

((marks >== 80) && (marks < 90))

((balance > 5000) || (no_of_trans > 25))

(!(grade == 'A'))

Branching: *The if Statement*
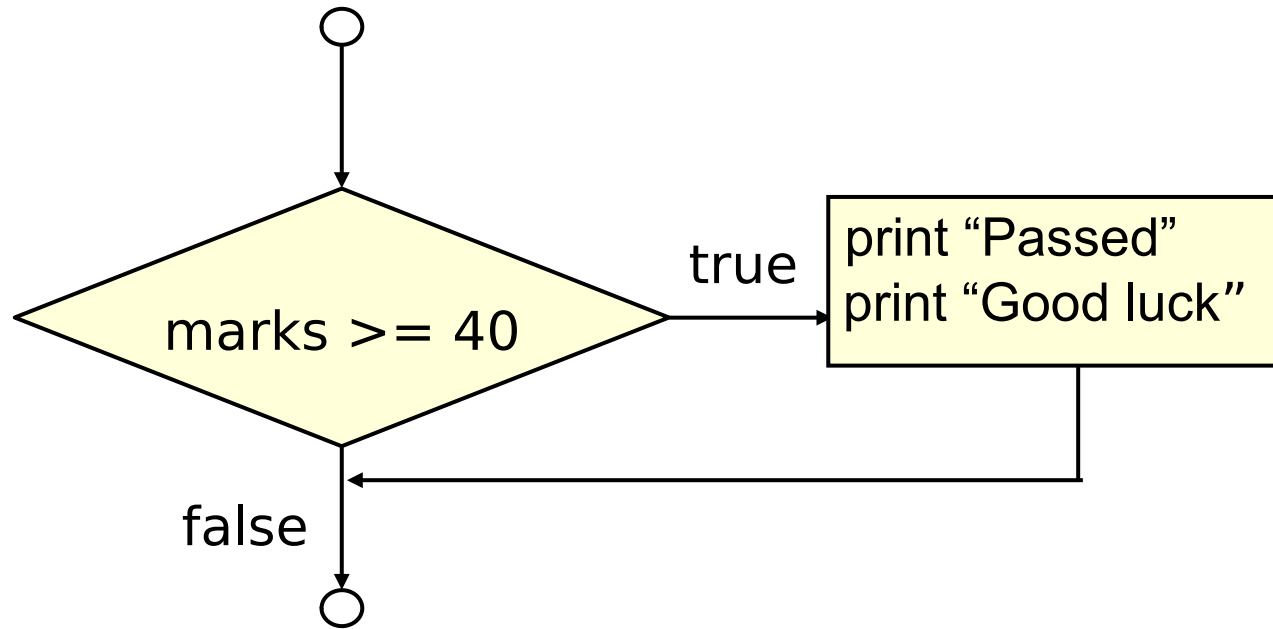
```
if (expression)

    statement;


if (expression)  {

    Block of statements

}
```

Note: indentation (after opening a brace, leave some horizontal space in the following lines, till you close that brace)

The condition to be tested is any expression enclosed in parentheses. The expression is evaluated, and if its value is non-zero, the statement is executed.

Note the position of the semi-colon.

A decision can be made on any expression.

zero :- false

nonzero :- true

```
if (marks>=40)  {

    printf("Passed \n");

    printf("Good luck\n");

}
printf ("End\n") ;
```

Branching: *if-else* *Statement*

```
if  (expression)  {

      Block of statements;

}

else  {

      Block of statements;

}
```

Note: indentation (after opening a brace, leave some horizontal space in the following lines, till you close that brace)

If a block contains only one statement, the braces are not needed

Branching: *if-else Statement*
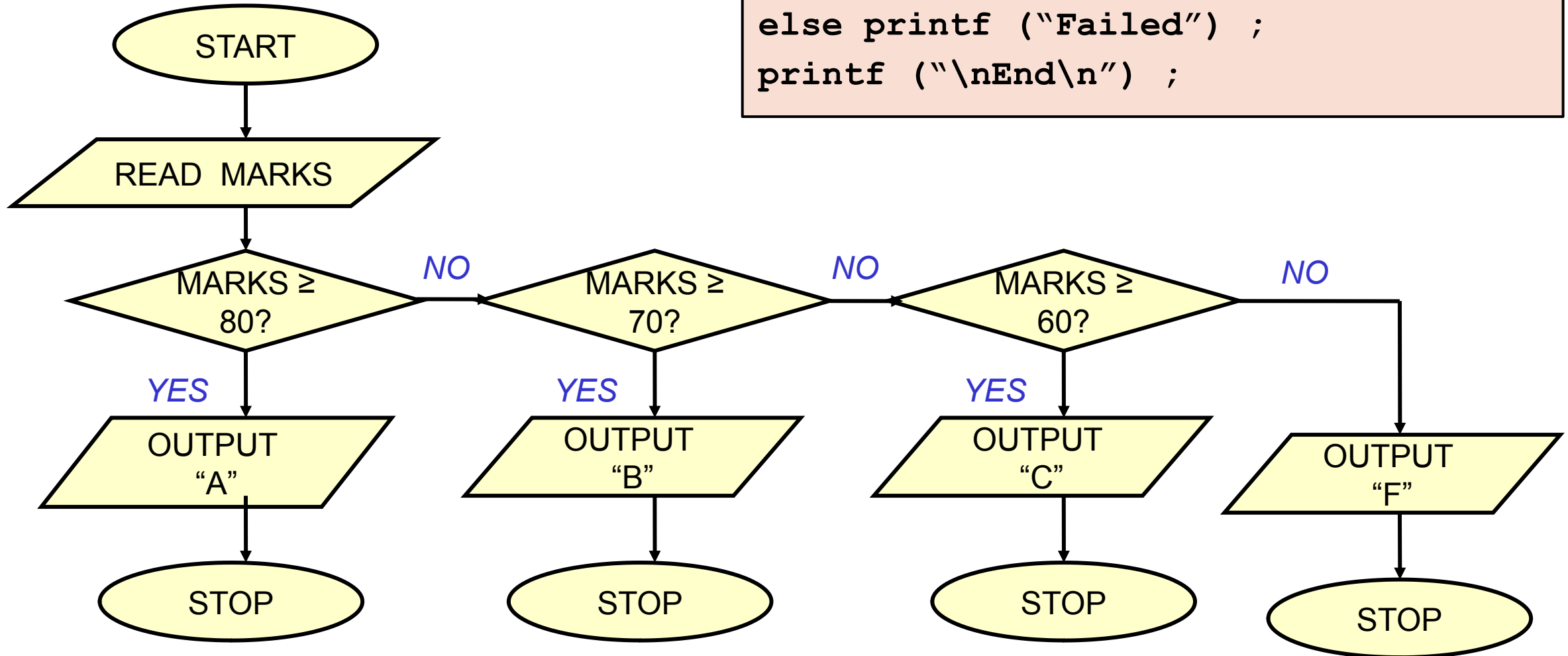
```
if  (expression)  {

        Block of statements;

}
else  {

        Block of statements;

}
```

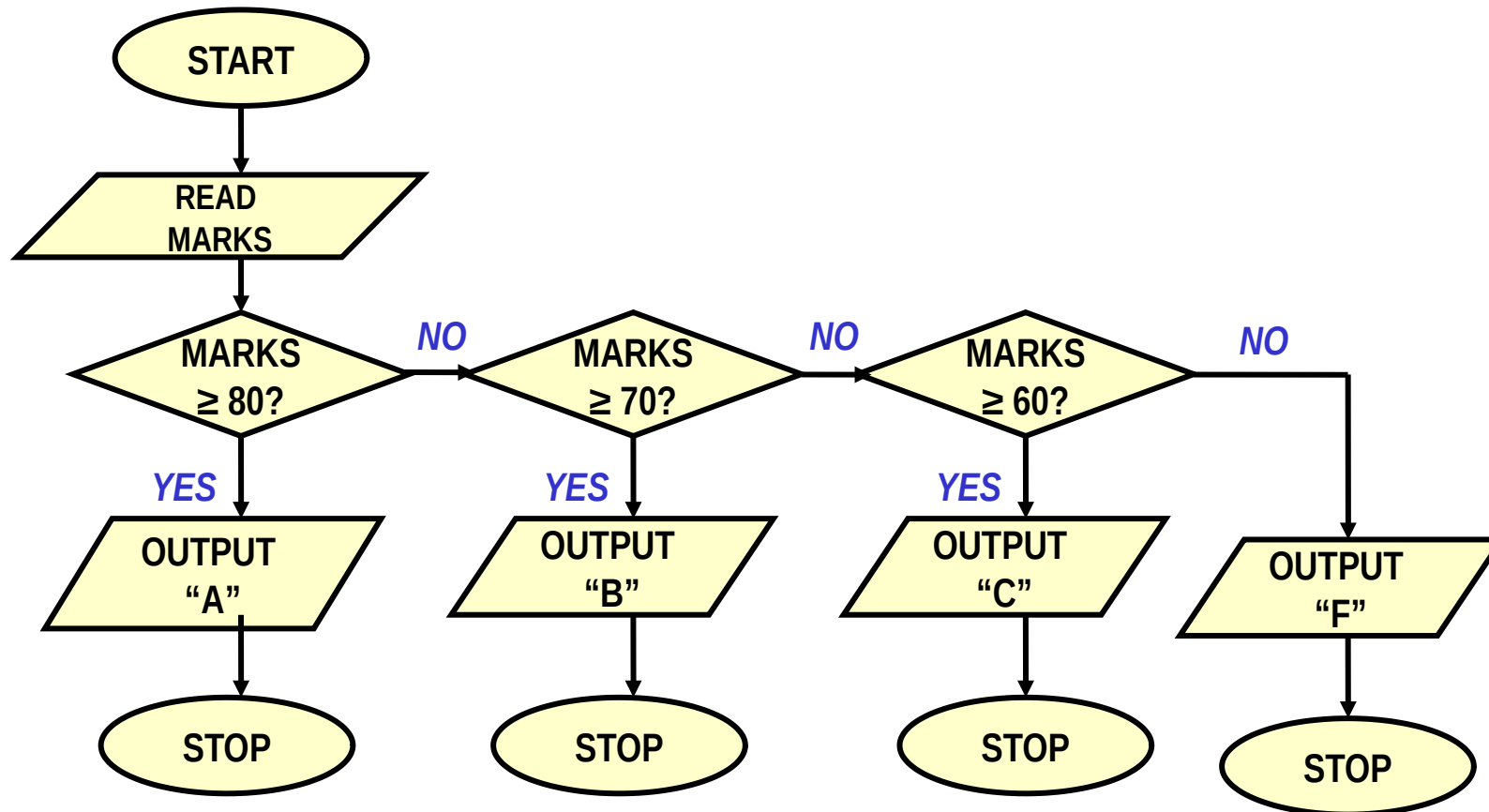If a block contains only one statement,
the braces are not needed

```
if (expression)  {

        Block of statements;

}

else if  (expression) {

        Block of statements;

}

else {

        Block of statements;

}
```

Grade Computation

```
if (marks >= 80) printf ("A") ;
else if (marks >= 70) printf ("B") ;
else if (marks >= 60) printf ("C") ;
else printf ("Failed") ;
printf ("\nEnd\n") ;
```
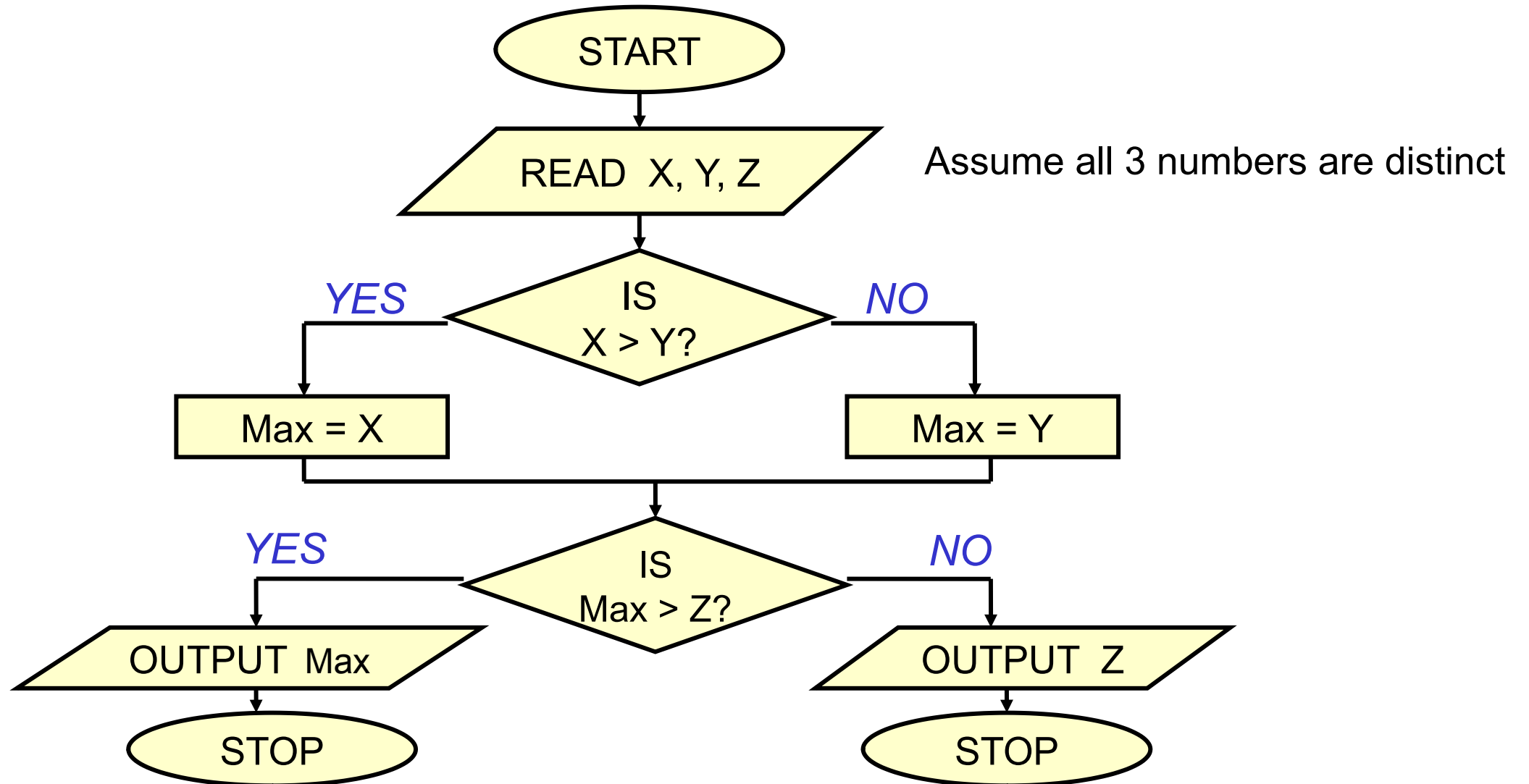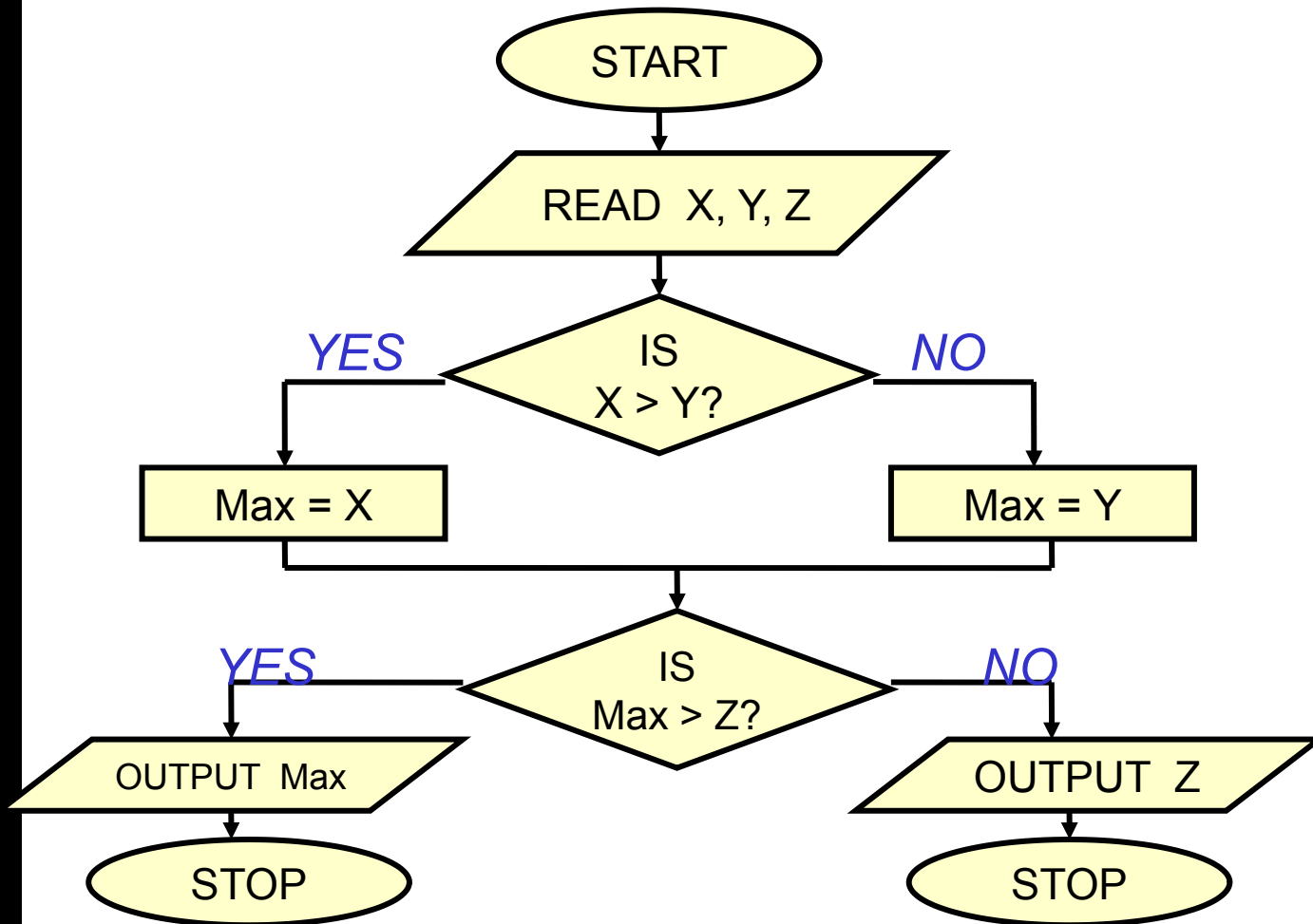
START

READ MARKS

MARKS ≥ 80?   NO   MARKS ≥ 70?   NO   MARKS ≥ 60?   NO

YES   YES   YES

OUTPUT "A"   OUTPUT "B"   OUTPUT "C"   OUTPUT "F"

STOP   STOP   STOP   STOP

Grade Computation

```c
int main ()
{
    int marks;
    scanf("%d", &marks);
    if (marks >= 80) {
        printf("A: ");
        printf("Good Job!");
    }
    else if (marks >= 70)
        printf("B ");
    else if (marks >= 60)
        printf("C ");
    else {
        printf("Failed: ");
        printf("Study Hard!");
    }
    return 0;
}
```

START

READ
MARKS

MARKS
≥ 80? →NO→ MARKS
≥ 70? →NO→ MARKS
≥ 60? →NO→

YES

OUTPUT
"A"

YES

OUTPUT
"B"

YES

OUTPUT
"C"

OUTPUT
"F"

STOP

STOP

STOP

STOP

Largest of three numbers

START

READ X, Y, Z

Assume all 3 numbers are distinct

IS X > Y?

YES → Max = X

NO → Max = Y

IS Max > Z?

YES → OUTPUT Max → STOP

NO → OUTPUT Z → STOP

```
int main () {
        int x, y, z, max;
        scanf ("%d%d%d",&x,&y,&z);
        if (x > y)
                max = x;
        else
                max = y;
        if (max > z)
                printf ("%d", max) ;
        else
                printf ("%d",z);
}
```

Another version

```c
int main ()  {
    int  a,b,c;
    scanf ("%d%d%d", &a, &b, &c);

    if ((a >= b) && (a >= c))
        printf ("\n The largest number is: %d", a);

    if ((b >= a) && (b >= c))
        printf ("\n The largest number is: %d", b);

    if ((c >= a) && (c >= b))
        printf ("\n The largest number is: %d", c);

    return 0;
}
```

Confusing Equality (==) and Assignment (=) Operators

Dangerous error

- **Does not ordinarily cause syntax errors.**
- **Any expression that produces a value can be used in control structures.**
- **Nonzero values are true, zero values are false.**

Example:

```
if ( payCode == 4 )
   printf( "You get a bonus!\n" );


if ( payCode = 4 )
   printf( "You get a bonus!\n" );   X
```

Dangling else problem

`if (exp1) if (exp2) stmta else stmtb`

```
if (exp1) {

    if (exp2)
            stmta
    else
            stmtb


}
```

OR

```
if (exp1)  {

    if (exp2)
            stmta
}
else
        stmtb
```

?

X
An "else" clause is associated with the closest preceding unmatched "if".

Which one is the correct interpretation?

Example – wrong program

Print "ABC" if a number is between 0 and 100, or "XYZ" if it is –ve. Do not print anything in other cases.

```c
int main()

{

    int x;

    scanf("%d", &x);

    if (x >= 0)

        if (x <= 100)

            printf("ABC\n");

    else

        printf("XYZ\n");

    return 0;

}
```

Outputs for different inputs

```
150
XYZ
```

Not what we want, should not have printed anything

```
-20
```

Not what we want, should have printed XYZ

Example - Correct Program

```c
int main()

{

    int x;

    scanf("%d", &x);

    if (x >= 0) {

        if (x <= 100)

            printf("ABC\n");

    }

    else

        printf("XYZ\n");

    return 0;

}
```

Outputs for different inputs

```
150

```

```
-20
XYZ
```

More examples

if e1 s1
else if e2 s2

if e1 s1
else if e2 s2
else s3

if e1 if e2 s1
else s2
else s3

if e1 if e2 s1
else s2

?

if e1 s1
else { if e2 s2 }

if e1 s1
else { if e2 s2
    else s3 }

if e1 { if e2 s1
    else s2 }
else s3

if e1 { if e2 s1
    else s2 }

While programming, it is always good to explicitly give the { and } to avoid any mistakes

The Conditional Operator ? :

This makes use of an expression that is either true or false. An appropriate value is selected, depending on the outcome of the logical expression.

Example:

interest =   (balance>5000) ? balance*0.2 : balance*0.1;

Returns a value

Equivalent to: if (balance > 5000) interest = balance * 0.2; else interest = balance * 0.1;

Equivalent also to: interest = balance * ((balance > 5000) ? 0.2 : 0.1);

More Examples

```
if ((a > 10) && (b < 5))
    x = a + b;
else
    x = 0;


x = ((a > 10) && (b < 5)) ? a + b : 0
```

```
if (marks >= 60)
    printf("Passed \n");
else
    printf("Failed \n");


(marks >= 60) ? printf("Passed \n") : printf("Failed \n");
```

The *switch* statement

This causes a particular group of statements to be chosen from several available groups.

- **Uses "switch" statement and "case" labels.**

```
switch (expression) {
    case const-expr-1: S-1
    case const-expr-2: S-2
        :
    case const-expr-m: S-m
    default: S
}
```

- **expression is any integer-valued expression**

- **const-expr-1, const-expr-2,...are any constant integer-valued expressions**
  - **Values must be distinct**

- **S-1, S-2, ...,S-m, S are statements / compound statements**

- **Default is optional, and can come anywhere (not necessarily at the end as shown), but put it at the end only (as shown)**

Behavior of switch
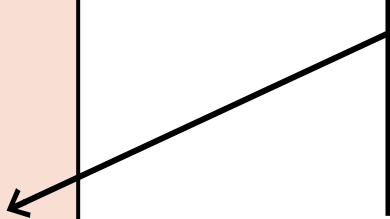
```
switch (expression)  {
        case const-expr-1: S-1
        case const-expr-2: S-2
         :
        case const-expr-m: S-m
        default: S
}
```

- **expression is first evaluated**

- **It is then compared with const-expr-1, const-expr-2,...for equality in order**

- **If it matches any one, all statements from that point till the end of the switch are executed (including statements for default, if present)**
  - **Use break statements if you do not want this (see example)**

- **Statements corresponding to default, if present, are executed if no other expression matches**

Examples

```
switch ( letter ) {

    case 'A':

        printf ("First letter \n");

        break;

    case 'Z':

        printf ("Last letter \n");

        break;

    default :

        printf ("Middle letter \n");

        break;

}
```

Will print this statement for all letters other than A or Z

Examples

```
switch ( choice = getchar( ) ) {
    case 'r' :
    case 'R' : printf("Red");
              break;
    case 'b' :
    case 'B' : printf("Blue");
              break;
    case 'g' :
    case 'G' : printf("Green");
              break;
    default:   printf("Black");
}
```

Since there isn't a **break** *statement here, the control passes to the next statement* (printf) *without checking the next condition.*

Another way

```
switch  ( choice = toupper(getchar()) )  {

    case 'R': printf ("RED \n");

              break;

    case 'G': printf ("GREEN \n");

              break;

    case 'B': printf ("BLUE \n");

              break;

    default:  printf ("Invalid choice \n");

}
```

Rounding a Digit

```
switch (digit)    {
        case 0:
        case 1:
        case 2:
        case 3:
        case 4: result = 0; printf ("Round down\n"); break;
        case 5:
        case 6:
        case 7:
        case 8:
        case 9: result = 10; printf("Round up\n"); break;
}
```

Example: checking if a character is a lower-case letter

```c
int main()

{

    char c1;

    scanf("%c", &c1);


    /* the ascii code of c1 must lie between the ascii codes of 'a' and 'z' */

    if (c1 >= 'a' && c1 <= 'z')

            printf ("%c is a lower-case letter\n", c1);

    else

            printf ("%c is not a lower-case letter\n", c1);

    return 0;

}
```

Example: converting a character from lowercase to uppercase

```c
int main()

{

    char c1;

    scanf("%c", &c1);

    /* convert to upper case if lower-case, else leave as it is */

    if (c1 >= 'a' && c1<= 'z')

        /* since ascii codes of upper-case letters are contiguous,
             the upper-case version of c1 will be as far away from
             the ascii code of 'A' as it is from the ascii code of 'a' */

        c1 = 'A' + (c1 - 'a');

    printf ("The letter is %c\n", c1);

    return 0;

}
```

Evaluating expressions

```c
int main () {
    int operand1, operand2;
    int result = 0;
    char operation ;
    /* Get the input values */
    printf ("Enter operand1 :");
    scanf("%d", &operand1) ;
    printf ("Enter operation :");
    scanf ("\n%c", &operation);
    printf ("Enter operand 2 :");
    scanf ("%d", &operand2);
    switch (operation)    {
    case '+' :
        result = operand1 + operand2;
        break;
```

```c
    case '-' :
        result = operand1 - operand2;
        break;
    case '*' :
        result = operand1 * operand2;
        break;
    case '/' :
        if (operand2 !=0)
            result = operand1 / operand2;
        else
            printf ("Divide by 0 error");
        break;
    default:
        printf ("Invalid operation\n");
        return;
    }
    printf ("The answer is %d\n", result);
    return 0;
}
```

Practice Problems

1. Read in 3 integers and print a message if any one of them is equal to the sum of the other two.

2. Read in the coordinates of two points and print the equation of the line joining them in y = mx +c form.

3. Read in the coordinates of 3 points in 2-d plane and check if they are collinear. Print a suitable message.

4. Read in the coordinates of a point, and the center and radius of a circle. Check and print if the point is inside or outside the circle.

5. Read in the coefficients a, b, c of the quadratic equation $ax^2 + bx + c = 0$, and print its roots nicely (for imaginary roots, print in x + iy form)

6. Suppose the digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 are mapped to the lowercase letters a, b, c, d, e, f, g, h, i, j respectively. Read in a single digit integer as a character (using %c in scanf) and print its corresponding lowercase letter. Do this both using switch and without using switch (two programs). Do not use any ascii code value directly.

7. Suppose that you have to print the grades of a student, with >= 90 marks getting EX, 80-89 getting A, 70-79 getting B, 60-69 getting C, 50-59 getting D, 35-49 getting P and <30 getting F. Read in the marks of a student and print his/her grade.