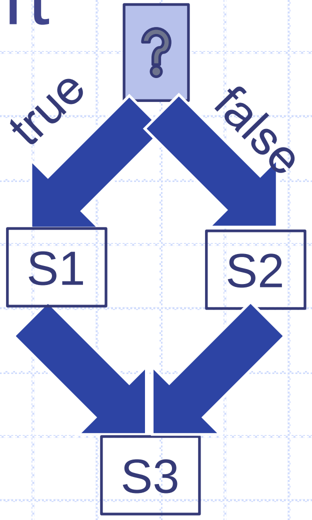
Conditional statements in C

- 3 types of conditional statements in C
  - if (cond) action
    else some-other-action
  - if (cond) action
  - switch-case
- Each action is a sequence of one or more statements!

if-else statement

- General form of the if-else statement

```
if (expression)
        statement S1
else
        statement S2
statement S3
```

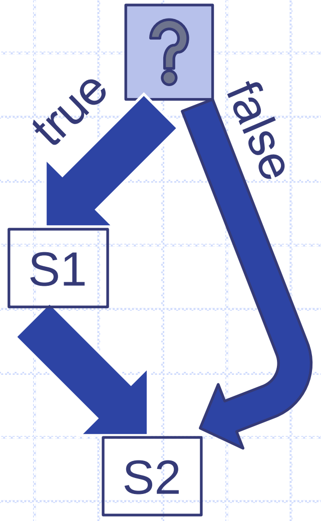


- Execution of if-else statement
  - First the expression is evaluated.
  - If it evaluates to a non-zero value, then S1 is executed and then control (program counter) moves to S3.
  - If expression evaluates to 0, then S2 is executed and then control moves to S3.
  - S1/S2 can be **block** of statements!

if statement (no else!)

- General form of the if statement

```
if (expression)
      statement S1
statement S2
```



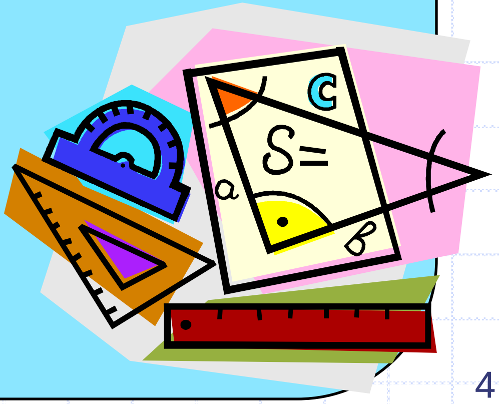
- Execution of if statement
  - First the expression is evaluated.
  - If it evaluates to a non-zero value, then S1 is executed and then control (program counter) moves to the statement S2.
  - If expression evaluates to 0, then S2 is executed.

Example

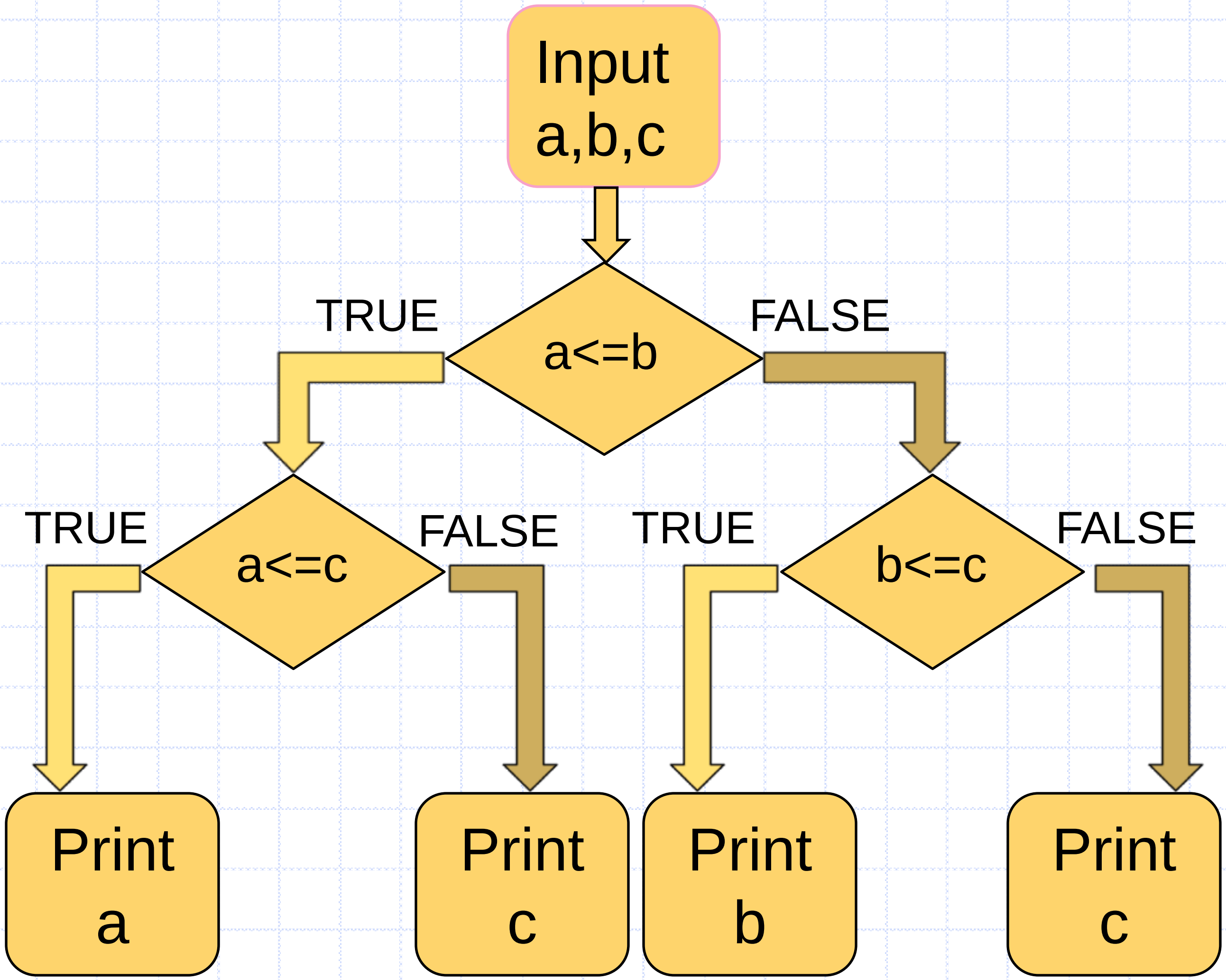
- Problem: Input a, b, c are real positive numbers such that c is the largest of these numbers. Print ACUTE if the triangle formed by a, b, c is an acute angled triangle and print NOT ACUTE otherwise.

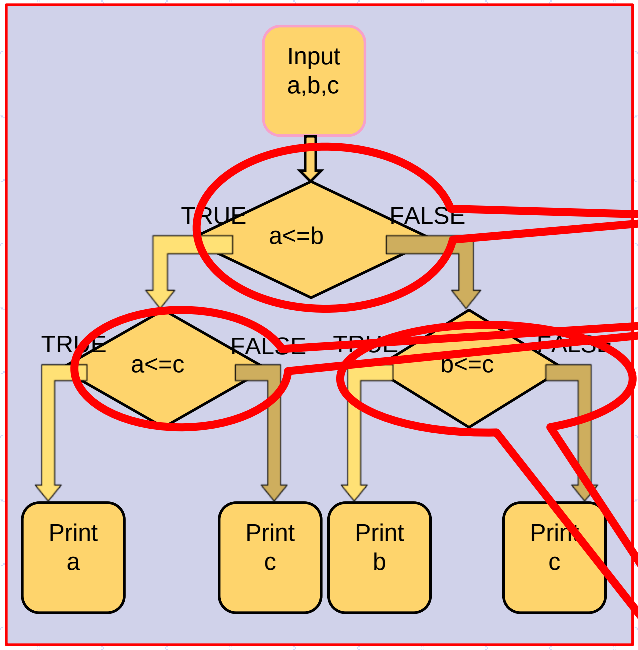
```
int main() {
        float a; float  b; float  c;
        scanf("%f%f%f", &a,&b,&c);        /* input a,b,c */

        if ( (a*a + b*b) >  (c*c) )  {  /* expression*/
            printf("ACUTE");
        }
        else {
           printf("NOT ACUTE");
        }
        return 0;
}
```



Finding min of 3 numbers





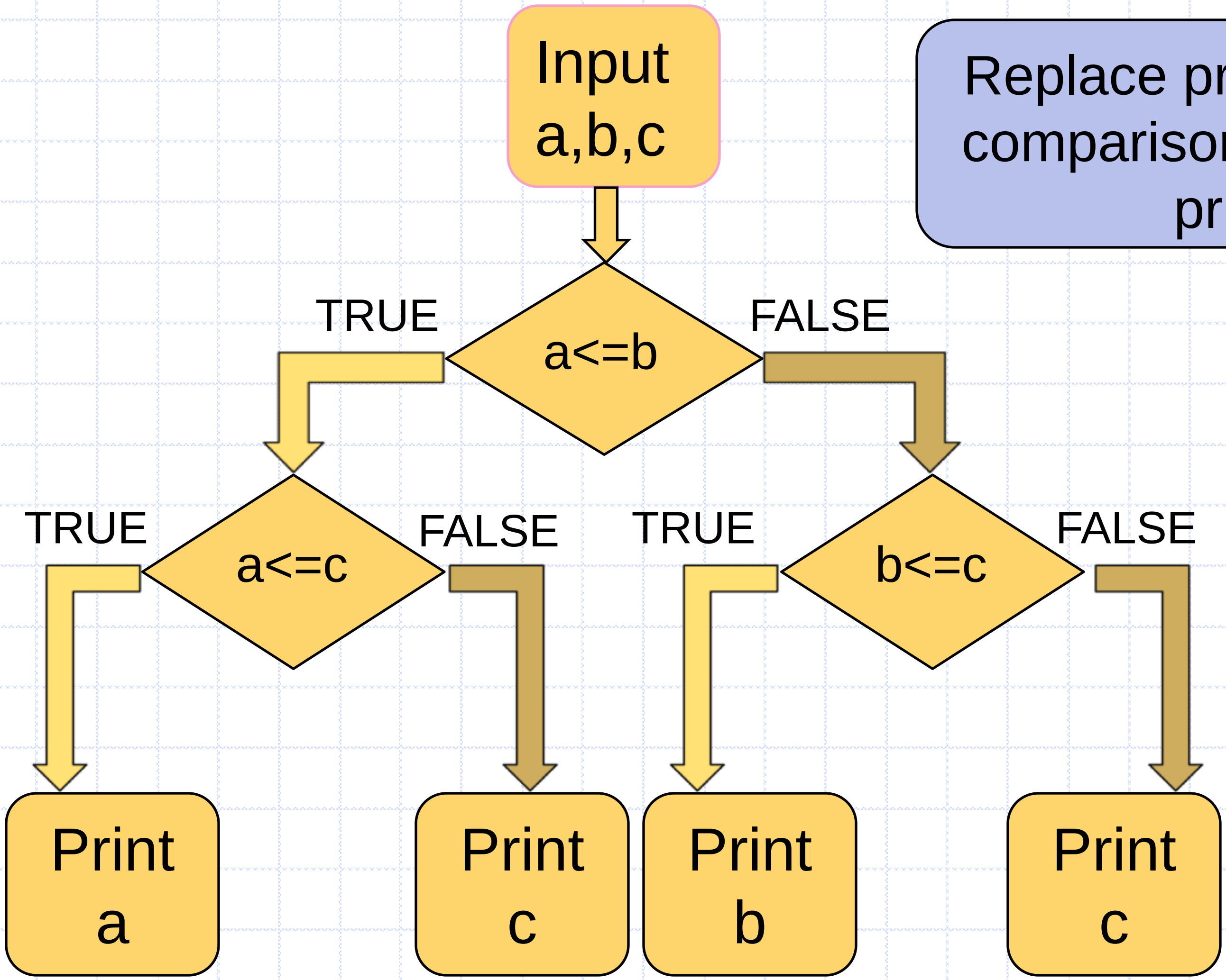
```
int a,b,c;
scanf(“%d%d%d”,&a,&b,&c);
if (a <= b) {
       if (a <= c) {
              printf(“min = %d”,a);
       }
       else {
              printf(“min = %d”, c);
       }
}
else {
        if (b <= c)  {
               printf(“min = %d”, b);
        }
        else {
               printf(“min =%d”, c);
        }
}
```

- Each branch translates to an if-else statement
- Hierarchical branches result in nested if-s

More Conditionals

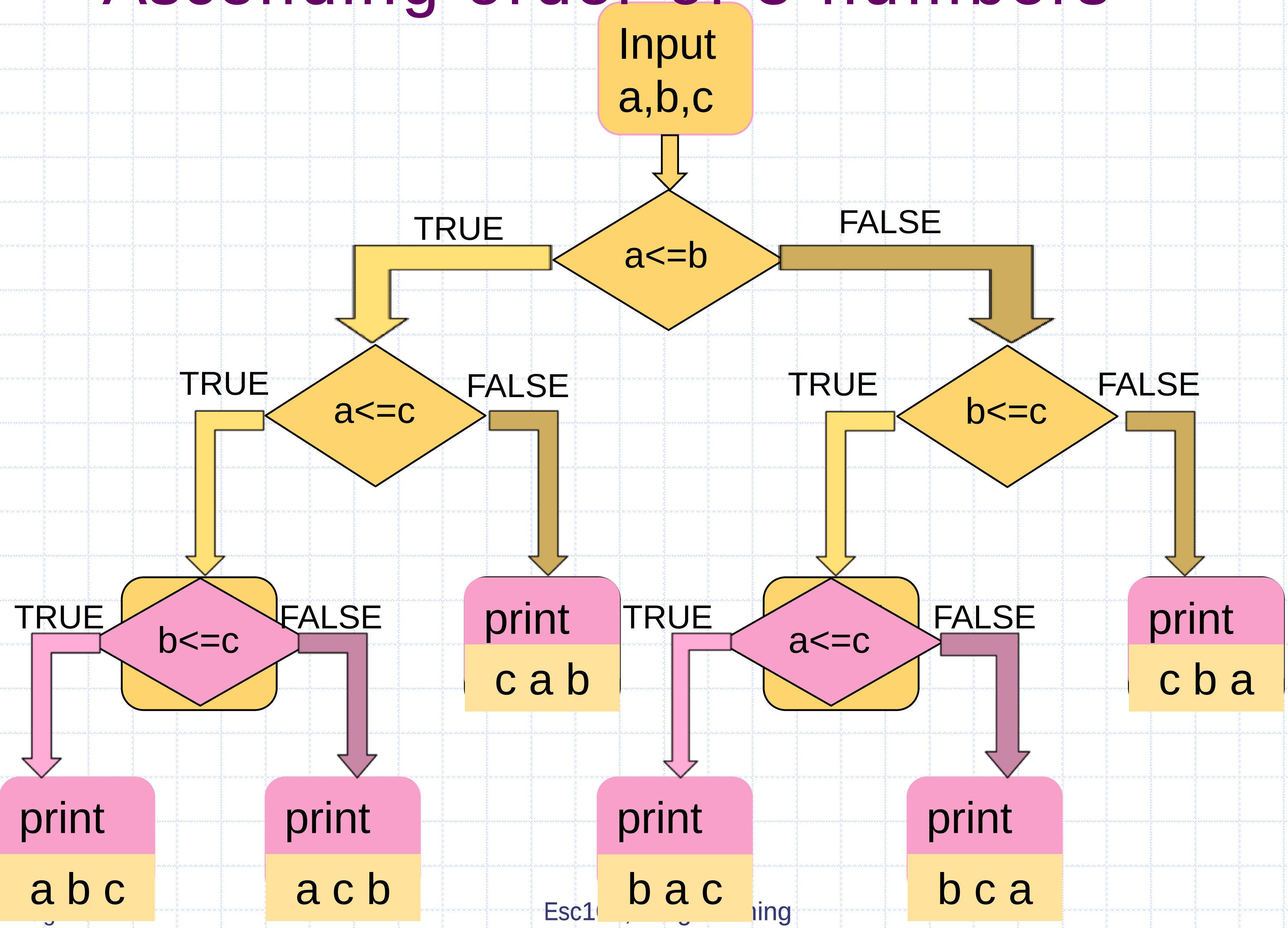
- Sorting a sequence of numbers (i.e., arranging the numbers in ascending or descending order) is a basic primitive.
- Problem: read three numbers into a, b and c and print them in ascending order.
  - Start with the flowchart for finding minimum of three numbers and add one more level of conditional check.
  - Then translate the flowchart into C program.

Finding min of 3 numbers



Replace print by more comparisons and then print.

Ascending order of 3 numbers



```
if (a <= b) {
    if (a <= c) {       /* a <= b and a <= c */
        if (b <= c) {   /* a <= b, a <= c, b <= c */
            printf("%d %d %d \n", a, b, c);
        } else {        /* a <= b, a <= c, c < b */
            printf("%d %d %d \n", a, c, b);
        }
    } else {            /* a <= b, c < a*/
        printf("%d %d %d \n", c, a, b) ;
    }
} else {                /* b < a */
    if (b <= c) {       /* b < a and b <= c */
        if (a <= c) {   /* b < a, b <= c, a <= c */
            printf("%d %d %d\n", b, a, c);
        } else {        /* b < a, b <= c, c < a */
            printf("%d %d %d\n", b, c, a); }
        }
    } else {            /* b < a, c < b */
        printf("%d %d %d\n", c, b, a); }
    }
}
```

Nested if, if-else

- Earlier examples showed us *nested* if-else statements

```
if (a <= b) {
        if (a <= c) { … }  else {…}
} else {
         if (b <= c)  { … } else { … }
}
```

- Because if and if-else are also statements, they can be used anywhere a statement or block can be used.

Else if

- A special kind of nesting is the chain of if-else-if-else-... statements

```
if (cond1) {
     stmt1
} else {
        if (cond2)  {
             stmt2
        } else {
            if (cond3) {
                 ….
            }
        }
}
```

General form of if-else-if-else...

```
if (cond1)
     stmt-block1
else if (cond2)
     stmt-block2
else if (cond3)
     stmt-block3
else if (cond4)
     stmt-block4
else if  …
else
      last-block-of-stmt
```

Example

- Given an integer day , where $1 \le day \le 7$, print the name of the weekday corresponding to day.

  1: Sunday

  2: Monday

  …

  7: Saturday

Printing the day

```
int day;
scanf (“%d”, &day);
if (day == 1) { printf(“Sunday”); }
else if (day == 2) { printf (“Monday”); }
else if (day == 3) { printf (“Tuesday”); }
else if (day == 4) { printf (“Wednesday”); }
else if (day == 5) { printf (“Thursday”); }
else if (day == 6) { printf (“Friday”); }
else if (day == 7) { printf (“Saturday”); }
else { printf (“ Illegal day %d”, day); }
```

Example 2

- Given an integer day , where $1 \le day \le 7$, print Weekday, if the day corresponds to weekday, print Weekend otherwise.

    1, 7: Weekend

    2,3,4,5,6: Weekday

Weekday - version 1

```
int day;
scanf (“%d”, &day);
if (day == 1) { printf(“Weekend”); }
else if (day == 2) { printf (“Weekday”); }
else if (day == 3) { printf (“Weekday”); }
else if (day == 4) { printf (“Weekday”); }
else if (day == 5) { printf (“Weekday”); }
else if (day == 6) { printf (“Weekday”); }
else if (day == 7) { printf (“Weekend”); }
else { printf (“ Illegal day %d”, day); }
```

Weekday - version 2

```
int day;
scanf (“%d”, &day);
if ((day == 1) || (day == 7)) {
      printf(“Weekend”);
} else if (  (day == 2) || (day == 3)
           || (day == 4) || (day == 5)
           || (day == 6)) {
      printf (“Weekday”);
} else {
      printf (“ Illegal day %d”, day);
}
```

Weekday - version 3

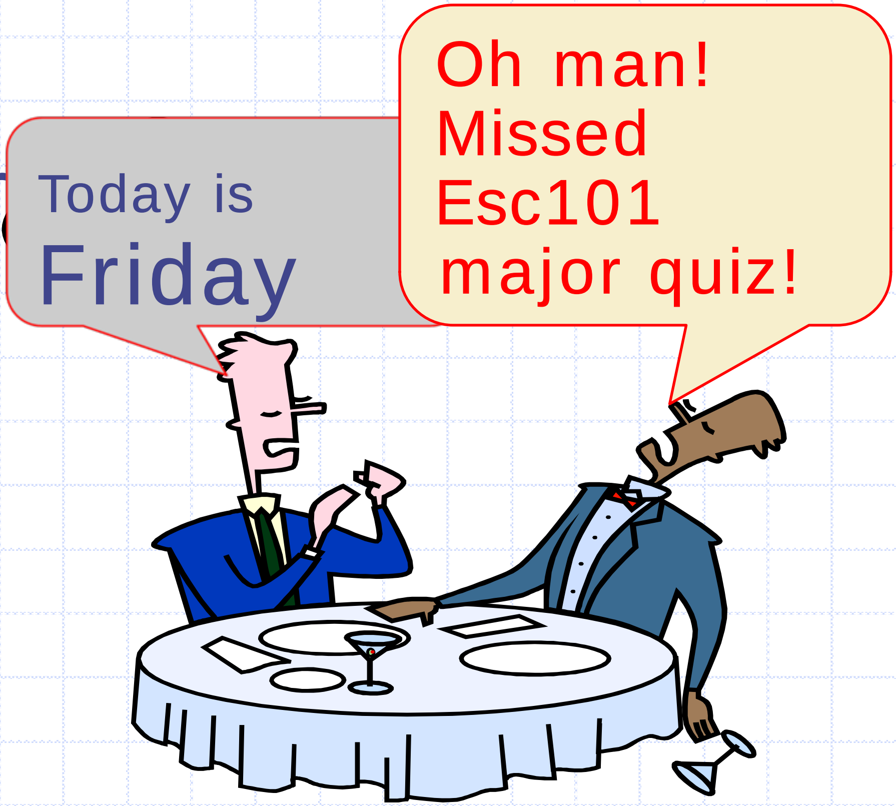
```
int day;
scanf (“%d”, &day);
if ((day == 1) || (day == 7)) {
        printf(“Weekend”);
} else if ( (day >= 2) && (day <= 6) ) {
        printf (“Weekday”);
} else {
        printf (“ Illegal day %d”, day);
}
```

Summary of if, if-else

- if-else, nested if's, else if.
- Braces {…} can be omitted if a block has only one statement.
- Multiple ways to solve a problem
  - issues of better readability
  - and efficiency.

Switch-case statement

- Multi-way decision
- Checks whether an expression matches one out of a number of constant integer (or char) values
- Execution *branches* based on the match found



Printing the day, version 2

```
switch (day) {
case 1: printf(“Sunday”); break;
case 2: printf (“Monday”); break;
case 3: printf (“Tuesday”); break;
case 4: printf (“Wednesday”); break;
case 5: printf (“Thursday”); break;
case 6: printf (“Friday”); break;
case 7: printf (“Saturday”); break;
default: printf (“ Illegal day %d”, day);
}
```

Weekday, version 4

```
switch (day) {
case 1:
case 7: printf (“Weekend”); break;
case 2:
case 3:
case 4:
case 5:
case 6: printf (“Weekday”); break;
default: printf (“ Illegal day %d”, day);
}
```

General Form of switch-case

```
switch (selector-expr) {
case label1: s1; break;
case label2: s2; break;
...
case labelN: sN; break;
default : sD;
}
```

Expr only of type INT
Execution starts at the matching case.

- **default** is optional. (= *remaining cases*)
- The location of **default** does not matter.
- The statements following a case label are executed one after other until a **break** is encountered (**Fall Through**)