Selection Statements
Notes

• Homework 4 is posted (due on Oct. 4, next Sunday)
Algorithms and Flow-chart
A typical programming task can be divided into two phases:

- **Problem solving phase**
  - produce an ordered sequence of steps that describe solution of problem
  - this sequence of steps is called an *algorithm*

- **Implementation phase**
  - implement the program in some programming language
Steps in Problem Solving

- First produce a general algorithm (one can use *pseudocode*).
- Refine the algorithm successively to get step by step detailed *algorithm* that is very close to a computer language.
- *Pseudocode* is an artificial and informal language that helps programmers develop algorithms. Pseudocode is very similar to everyday English.
Steps in Problem Solving

How Google search engine ranks the results?
What is the algorithm behind Google’s PageRank algorithm?

How Netflix recommends movies?
What is the algorithm behind Netflix recommendation algorithm?

How Facebook recommends friends?
What is the algorithm behind Facebook’s suggestions?

We are not going to solve these problems in this course, just to see how important the first would be!
Example 1:

Write an algorithm to determine a student’s final grade and indicate whether it is passing or failing. The final grade is calculated as the average of four marks.
Pseudocode & Algorithm

Pseudocode:

- **Input** a set of 4 marks
- **Calculate** their average by summing and dividing by 4
- if average is below 50
  - *Print* “FAIL”
- else
  - *Print* “PASS”

**Question:** if user mistakenly enters a negative number as input mark!
Detailed Algorithm

- Step 1: Input M1, M2, M3, M4
- Step 2: GRADE ← (M1 + M2 + M3 + M4) / 4
- Step 3: if (GRADE < 50) then
  Print “FAIL”
  else
  Print “PASS”
endif
The Flowchart

- A **flowchart** is a type of diagram that represents an algorithm, workflow or process, showing the steps as boxes of various kinds, and their order by connecting them with arrows.

- (Technical) A **graphical** representation of the sequence of operations in an information system or program.
  - shows logic of an algorithm
  - emphasizes individual steps and their interconnections, e.g. control flow from one action to the next.

The big bang theory :-)

The big bang theory :-)
Flowchart Symbols

The following are some of the commonly used shapes used in flowcharts. Generally, flowcharts flow from top to bottom and left to right.

<table>
<thead>
<tr>
<th>Name</th>
<th>Symbol</th>
<th>Use in Flowchart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oval</td>
<td><img src="image" alt="Oval" /></td>
<td>Denotes the beginning or end of the program</td>
</tr>
<tr>
<td>Parallelogram</td>
<td><img src="image" alt="Parallelogram" /></td>
<td>Denotes an input operation</td>
</tr>
<tr>
<td>Rectangle</td>
<td><img src="image" alt="Rectangle" /></td>
<td>Denotes a process to be carried out e.g. addition, subtraction, division etc.</td>
</tr>
<tr>
<td>Diamond</td>
<td><img src="image" alt="Diamond" /></td>
<td>Denotes a decision (or branch) to be made. The program should continue along one of two routes. (e.g. IF/THEN/ELSE)</td>
</tr>
<tr>
<td>Hybrid</td>
<td><img src="image" alt="Hybrid" /></td>
<td>Denotes an output operation</td>
</tr>
<tr>
<td>Flow line</td>
<td><img src="image" alt="Flow line" /></td>
<td>Denotes the direction of logic flow in the program</td>
</tr>
</tbody>
</table>
Step 1: Input $M_1,M_2,M_3,M_4$
Step 2: $\text{GRADE} \leftarrow (M_1+M_2+M_3+M_4)/4$
Step 3: if $(\text{GRADE} < 50)$ then
    Print “FAIL”
else
    Print “PASS”
endif
Example:

Write an algorithm and draw a flowchart that will read the two sides of a rectangle and calculate its area.

Pseudocode
Example:

Write an algorithm and draw a flowchart that will read the two sides of a rectangle and calculate its area.

Pseudocode

Step 1- Input the width (W) and Length (L) of a rectangle
Step 2- Calculate the area (A) by multiplying L with W
Step 3- Print A
Algorithm

- Step 1: Input $W, L$
- Step 2: $A \leftarrow L \times W$
- Step 3: Print $A$

How to avoid negative numbers?
Algorithms and Flow-chart

START

Input W, L

L >= 0

W >= 0

Y

A = W * L

Print A

N

N

Print Error!

STOP
Write an algorithm and draw a flowchart that will calculate the roots of a quadratic equation
\[ ax^2 + bx + c = 0 \]

**Input:** a, b, and c

**Output:** x1 and x2 (the roots)

\[ x1 = 0.5, \ x2 = -5 \]
Write an algorithm and draw a flowchart that will calculate the roots of a quadratic equation

\[ ax^2 + bx + c = 0 \]

\[
x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}.
\]

\[
x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}.
\]
Pseudocode:

- **Input the coefficients** \((a, b, c)\) **of the quadratic equation**
- **Calculate** \(d\) \[ \quad d = \sqrt{b^2 - 4ac} \]
- **Calculate** \(x_1\)
- **Calculate** \(x_2\)
- **Print** \(x_1\) and \(x_2\)
**Algorithm:**

- **Step 1:** Input a, b, c
- **Step 2:** $d \leftarrow \sqrt{b \times b - 4 \times a \times c}$
- **Step 3:** $x_1 \leftarrow (-b + d) / (2 \times a)$
- **Step 4:** $x_2 \leftarrow (-b - d) / (2 \times a)$
- **Step 5:** Print $x_1$, $x_2$
**Algorithms and Flow-chart**

- **Algorithm:**
  - **Step 1:** Input $a$, $b$, $c$
  - **Step 2:** $d \leftarrow \sqrt{b \times b - 4 \times a \times c}$
  - **Step 3:** $x_1 \leftarrow (-b + d) / (2 \times a)$
  - **Step 4:** $x_2 \leftarrow (-b - d) / (2 \times a)$
  - **Step 5:** Print $x_1$, $x_2$

---

What if $b^2 - 4ac$ be negative?

$\sqrt{\text{negative}}$ not defined!
Algorithms and Flow-chart

What if $b^2 - 4ac$ be negative?

\[ \sqrt{\text{negative}} \quad \text{not defined!} \]

\[ \Delta > 0 \quad \text{two roots} \]
\[ \Delta = 0 \quad \text{single root} \]
\[ \Delta < 0 \quad \text{no root at all} \]
Example: Finding Minimum

Write an algorithm and draw a flowchart that will read four numbers and finds the minimum number among them?

**Input:** a, b, c, d

**Output:** min(a, b, c, d)

Input: 7, 3, 32, 8

Output: 3
Example: Sorting

Write an algorithm and draw a flowchart that will read four numbers and sort the numbers?

**Input:** a, b, c, d

**Output:** input numbers in ascending order

```
4, 9, 3, 7
```

```
3, 4, 7, 9
```
Write an algorithm and draw a flowchart that will read a number in decimal and converts it to binary?

**Input**: a

**Output**: input number in binary system

23 $\rightarrow$ 00010111
Control Structures in C
Control Structures:

A combination of individual instructions into a single logical unit with one entry point and one exit point

Control the flow of execution in a program

**Compound Statement:** a group of statements bracketed by `{` and `}` that are executed sequentially
There are three kinds of execution flow:

**Sequence** [built-in to C]:
Unless otherwise directed, one statement after the next is executed.

**Selection** [three types]:
Depending on a condition, *select* between one statement or another.

**Repetition** [three types]:
Depending on a condition, execute one or more statements *repeatedly*.
Control Structures in C

Sequence

Selection

Repetition (loop)
Selection Structures
Selection Structures in C: Overview

Three kinds of selections structures

- **if** (also called, ‘single-selection’)
  - if *condition* is true
    - Perform action
  - if *condition* is false, action is skipped, program continues

- **if/else** (also called, ‘double-selection’)
  - if *condition* is true
    - Perform action
  - else (if *condition* is false)
    - Perform a *different* action (this will be skipped if condition is true)

- **switch** (also called ‘multiple-selection’)
  - Allows selection among many actions depending on the integral value of a variable or expression
Single Selection IF - Flowchart

Speed > 65

TRUE

Print “You’re speeding”

FALSE

connector

flow line

decision symbol

action symbol
IF-Else Selection- Flowchart

can be done by using two single-IF selections!
A program may choose among alternative statements by testing the value of key variables.
  e.g., if( your_grade > 60 )
      printf("you are passed!");

- **Condition** is an expression that is either false (represented by 0) or true (represented by 1).
  e.g., “your_grade > 60” is a condition.

- Conditions may contain **relational** or **equality operators**, and have the following forms.
  variable **relational-operator** variable (or constant)
  variable **equality-operator** variable (or constant)
If Statement
If Statement

\[
\text{if ( expression ) statement}
\]

•() required around expression
•if: lowercase
•If expression evaluates to non zero, then statement is executed

```c
int answer;
printf(“Guess my lucky number:\n”);
scanf(“%d”, &answer);
if (answer == 5)
    printf(“You are correct!\n”);
printf(“We are done”);
```
Compound Statement

```c
if ( expression ) { statements }
```

• Use {} to execute multiple statements
• Compound statement can be written on one line

```c
int answer;
printf("Guess my lucky number:\n");
scanf("%d", &answer);
if (answer == 5)
{
    printf("You are correct!\n");
    printf("How did you know?\n");
}
printf("We are done");
```
if else statement

if (boolean_expression)
  yes_statements
else
  no_statements

• else statements are executed if expression evaluates to 0
Example code:

```c
int answer;
printf("Guess my lucky number:\n");
scanf("%d", &answer);
if (answer == 5)
    printf("You are correct!\n");
else
    printf("You are wrong!\n");
```

- **else statements** are executed if expression evaluates to 0.
Cascading if statements

if (delta < 0)
    printf("No real roots\n");
else
    if (delta == 0)
        printf("Exactly one real root\n");
    else
        printf("Two distinct roots\n");

Use when:
• series of conditions
• stop as soon as one if true
Cascading if statements

```
if (delta < 0)
    printf("no roots");
else if (delta == 0)
    printf("one root");
else
    printf("two roots");
```
Cascading if statements

```c
if (delta < 0)
    printf("No real roots\n");
else
    if (delta == 0)
        printf("Exactly one real root\n");
    else
        printf("Two distinct roots\n");
```

Diagram:

- If `delta < 0`, print "No real roots".
- If `delta == 0`, print "Exactly one real root".
- If `delta > 0`, print "Two distinct roots".
Cascading if statements

if (delta < 0)
  printf("No real roots\n");
else
  if (delta == 0)
    printf("Exactly one real root\n");
  else
    printf("Two distinct roots\n");