TOPIC 1
REVIEW OF C++ BASICS

Data types and Selection
Variables are locations in memory
- Their values are represented by 1’s and 0’s
- The format and length of the binary code determines the kind of data the variable stores

Data fall into several general categories
- Whole numbers (0, 1, -8, 10023)
- Real Numbers (1.0, -6.7, 10e-5)
- Characters (‘a’, ‘B’, “hello world”)
- Boolean Values (true or false)
Boolean Data

- Most data types are familiar:
  - Integers, Double, Char

- Boolean data is limited to having one of two values: true or false

- Boolean values are used when making decisions in programs
C++ allows us to conditionally execute blocks of code

```cpp
if ( <boolean expression> ) {
    statement 1;
    statement 2;
    ...
}
```

statements inside if block only execute if the Boolean expression evaluates to true
If / else blocks

- Often, we want to do one of two different things, based on a condition
  - Ask the user to enter the sum of randomly generated numbers:
  - If correct, congratulate them
  - Otherwise (else), display the correct answer

```cpp
if ( input == correctAnswer ) {
    cout << “Good Job!” << endl;
}
else {
    cout << “Sorry, correct answer is “ << correctAnswer << endl;
}
```
We can string together multiple if statements with the `else` command:

```java
if ( <boolean expression> ) {
    ...
}
else if ( <boolean expression> ) {
    ...
}
else {
    ....
}
```

you can have several else if blocks
Proper indentation of code is required

We adopt the following indentation standard (no exceptions!)

```cpp
int main() {
  int a, b;
  // read a and b from user...
  if (a == b) {
    cout << "These numbers are equal" << endl;
  }
  else if (a < b) {
    cout << "A is smaller" << endl;
  }
  else {
    cout << "B is smaller" << endl;
  }
  cout << "Thank you" << endl;
}
```

Matching brackets always on same column

Each nested bracket group indents one additional tab

Closing brackets on new line
Logical Operators

\(<\text{Boolean expression}> \&\& \ <\text{Boolean expression}> \rightarrow \text{Boolean value}\)

\(<\text{Boolean expression}> \mid \mid \ <\text{Boolean expression}> \rightarrow \text{Boolean value}\)

\(! \ <\text{Boolean expression}> \rightarrow \text{Boolean value}\)
# Truth Tables

<table>
<thead>
<tr>
<th><strong>AND</strong></th>
<th><strong>A</strong></th>
<th><strong>B</strong></th>
<th><strong>A &amp;&amp; B</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>TRUE</td>
<td></td>
<td>TRUE</td>
</tr>
<tr>
<td>TRUE</td>
<td>FALSE</td>
<td></td>
<td>FALSE</td>
</tr>
<tr>
<td>FALSE</td>
<td>TRUE</td>
<td></td>
<td>FALSE</td>
</tr>
<tr>
<td>FALSE</td>
<td>FALSE</td>
<td></td>
<td>FALSE</td>
</tr>
</tbody>
</table>

| **OR**  | **A** | **B** | **A || B** |
|---------|-------|-------|------------|
| TRUE   | TRUE |      | TRUE       |
| TRUE   | FALSE|      | TRUE       |
| FALSE  | TRUE |      | TRUE       |
| FALSE  | FALSE|      | FALSE      |

<table>
<thead>
<tr>
<th><strong>NOT</strong></th>
<th><strong>A</strong></th>
<th><strong>!A</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>FALSE</td>
<td></td>
</tr>
<tr>
<td>FALSE</td>
<td>TRUE</td>
<td></td>
</tr>
</tbody>
</table>
Operator Precedence

Parenthesis

!, - (negation)

*, / , %

+, -

<, <=, >=, >

==, !=

&&

||

=

Highest Precedence

Lowest Precedence
switch (IntegerExpression) {
    case IntegerConstant:
        // statements
        break;
    case IntegerConstant:
        // statements
        break;
    ...  
    default:
        // execute if no cases are a match
}
If and Switch Relationship

```cpp
if (x == 5) {
    cout << "1" << endl;
}
else if (x == 6) {
    cout << "2" << endl;
}
else if (x == 7) {
    cout << "3" << endl;
}
else {
    cout << "The rest" << endl;
}
```

```cpp
switch (x) {
    case 5:
        cout << "1" << endl;
        break;
    case 6:
        cout << "2" << endl;
        break;
    case 7:
        cout << "3" << endl;
        break;
    default:
        cout << "The rest" << endl;
}
```
Rules when using Switch

- Be *extremely* careful to not omit `break` when it should be there!
  - There are times omitting it is actually quite useful however.
  - If *omitted*, always add comment explaining why.

- You cannot compare two *variables* with a switch - you can only compare against *integer or character literals*
  - Booleans also work, but not that useful
Example: Area Calculator

- Ask the user if they want to compute the area of a Rectangle (r), Square (s), or Circle (c).
  - For Rectangle, ask for width and height.
  - For Square, ask for width only.
  - For Circle ask for radius.

- Support upper and lower case.