CSE 115
Introduction to Computer Science I
Road map

▶︎ Review ◀

functions and control flow exercises
Selection

if statement

parentheses are required

```java
if ( expression ) {
  statement ;
  statement ;
  ...
  statement ;
}
```
Selection

if statement

a code block ('then' clause)

if ( expression ) {
    statement ;
    statement ;
    ...
    statement ;
}

Selection

if-else statement

else does not take an expression

if ( expression ) {
  statement ;
  statement ;
  ...
  statement ;
}
else {
  statement ;
  statement ;
  ...
  statement ;
}
there is no `elif` keyword

```java
if ( expression ) {
   ...
}
else if ( expression ) {
   ...
}
else if ( expression ) {
   ...
}
else {
   ...
}
```
Road map

Review

- functions and control flow exercises
Define a function named `totalCost` that takes two values (a price per item and the number of items) and returns the total cost for the indicated number of items.

Discuss with your neighbors what the inputs to and output from this function are.

Write out some test cases for this function.

Define the function in JavaScript.
/* Test cases:
    totalCost(2,3) should return 6
    totalCost(3.5,6) should return 21
    totalCost(19,0) should return 0
*/

// Possible definition for function
function totalCost(ppi, n) {
    return ppi * n;
}

Define a function named `shipping` that takes one value (the total pre-tax amount of an order) and returns the shipping cost. Assume that orders of at least $100 get free shipping, but that below $100 shipping costs 10% of the pre-tax order amount.

Discuss with your neighbors what the inputs to and output from this function are.

Write out some test cases for this function.

Define the function in JavaScript.
shipping

/* Test cases:
   shipping(110) should return 0
   shipping(100) should return 0
   shipping(99) should return 9.9
   shipping(50) should return 5
*/

// Possible definition for function
function shipping(total) {
    if (total>=100) {
        return 0;
    } else {
        return 0.1 * total;
    }
}
Define a function named `invoice` that takes two values (a price per item and the number of items) and returns the total invoice amount for an order. The order total includes the total cost of the items ordered, plus shipping, plus 8% tax on the pre-shipping total.

Discuss with your neighbors what the inputs to and output from this function are.

Write out some test cases for this function.

Define the function in JavaScript.
/ * Test cases: 
  invoiceTotal(5,20) should return 108 
  invoiceTotal(5,10) should return 59 
  invoiceTotal(10,2) should return 23.6 
 */

// Possible definition for function:
function invoiceTotal(ppi, n) {
  var orderAmount = totalCost(ppi,n);
  var shippingAmount = shipping(orderAmount);
  var taxAmount = orderAmount * 0.08;
  var total = orderAmount + shippingAmount + taxAmount;
  return total;
}
LDL cholesterol

LDL (low density-lipoprotein) cholesterol is also called "bad" cholesterol. LDL can build up on the walls of your arteries and increase your chances of getting heart disease. If you do not have heart or blood vessel disease and are not at high risk for developing heart disease, the following guidelines apply.

Your LDL cholesterol number is:

- **Optimal** if it is less than 100.
- **Near optimal/above optimal** if it is 100-129.
- **Borderline high** if it is 130-159.
- **High** if it is 160-189.
- **Very high** if it is 190 or above.

https://my.clevelandclinic.org/health/articles/11920-cholesterol-numbers-what-do-they-mean

Define a function named `ldl_level` that takes an LDL level as input and returns the corresponding category: optimal, near optimal, borderline high, high, or very high.

Discuss with your neighbors, write test cases, and then define the function in JavaScript.
LDL cholesterol

/* Test cases:
   ldl_level(99) should return "optimal"
   ldl_level(100) should return "near optimal"
   ldl_level(129) should return "near optimal"
   ldl_level(130) should return "borderline high"
   ldl_level(159) should return "borderline high"
   ldl_level(160) should return "high"
   ldl_level(189) should return "high"
   ldl_level(190) should return "very high"
*/

// Possible definition of function:
function ldl_level(ldl) {
    if (ldl < 100) { return "optimal"; }
    else if (ldl < 130) { return "near optimal"; }
    else if (ldl < 160) { return "borderline high"; }
    else if (ldl < 190) { return "high"; }
    return "very high";
}
In case class finishes other questions
Define a function that computes the arithmetic mean of three values.

Q: How might we approach this problem?

Brainstorm with your neighbors. Be sure to agree on the formula for an arithmetic mean. Be prepared to explain (in English, not code) your approach. If you write code too, that's cool.
arithmetic mean

/* Test cases - add your own!
   a_mean(1,2,9) should return 4
*/

// Possible definition of function
function a_mean( x , y , z ) {
    return ( x + y + z ) / 3;
}
Define a function that computes the geometric mean of three values.

Q: How might we approach this problem?

Brainstorm with your neighbors. Be sure to agree on the formula for an geometric mean. Be prepared to explain (in English, not code) your approach. If you write code too, that's cool.
geometric mean

/* Test cases - add your own!
   g_mean(1,3,9) should return 3
*/

// Possible definition of function
function g_mean(x, y, z) {
    return (x * y * z)**(1/3);
}
Define a function that computes either the arithmetic or geometric mean of three values, based on whether the fourth argument to the function is 0 or 1.

0 --> arithmetic mean
1 --> geometric mean

Q: How might we approach this problem?

Brainstorm with your neighbors. Be sure to agree on the formula for an arithmetic mean. Be prepared to explain (in English, not code) your approach. If you write code too, that's cool.
choice of mean

/* Test cases */

// Possible definition of function
function mean( x , y , z , type ) {
    if ( type == 0 ) {
        return a_mean(x, y, z);
    }
    else if ( type== 1 ) {
        return g_mean(x, y, z);
    }
}

Parameterize the type of mean to compute: no change in program is needed.
mean(a,b,c,t) vs a_mean(a,b,c) g_mean(a,b,c)

Same function is called, but with different arguments.

The name of a function is fixed at runtime.
The values of arguments can vary at runtime.