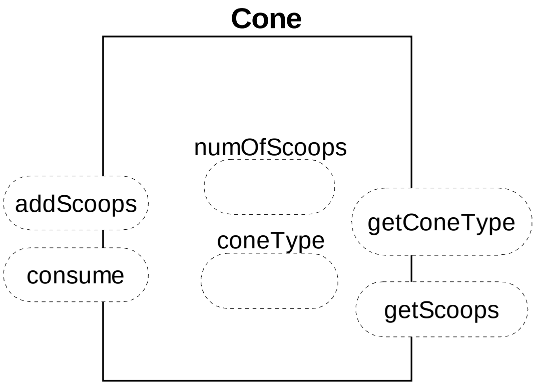
WEB-BASED APPLICATIONS
MIDTERM 1
SAMPLE QUESTIONS

Write the code for the instantiable class Cone that represents an ice cream cone. The following information is provided. *If something isn't specified, do something reasonable.*



`Cone(t, c)`
constructs an ice cream cone with **c** number of scoops and cone type of **t**.

`addScoops(n)`
adds **n** scoops to the ice cream cone.

`getScoops( )`
returns the number of scoops of the ice cream cone.

`consume()`
sets the number of scoops to 0 and the cone type to -1.

`getConeType()`
returns the type of the cone: 1 for wafer, 2 for sugar.

```
function Cone(t, c) {

      // private variables
      var numOfScoops;
      var coneType;

      // assign properties-values to private variables
      numOfScoops = c;
      coneType =t;

      // public method for adding "n" scoops
      this.addScoops = function(n) {
            numOfScoops = numOfScoops + n;
      }

      // public method for returning number of scoops
      this.getScoops = function() {
            return numOfScoops;
      }

      // public method for consuming ice-cream cone
      this.consume = function() {
            numOfScoops = 0;
            coneType= -1;
      }

      // public method for getting the type of cone
      this.getConeType = function() {
            return coneType;
      }
} /* end of Class definition for Cone */
```

You have to ask yourself is how do we declare and use the variables and the method.

Let’s say Mark and Rob, two buddies decide that they want to get ice cream. Mark wants 3 scoops on a sugar (2) cone and Rob wants 1 scoop on a wafer (1) cone. To create these instance of ice cream:

```
var MarkIceCream = new Cone(2, 3);
var RobIceCream = new Cone(1, 1);
```

After seeing how delicious the ice cream is, Rob decides that he wants one more scoop. To do so you need to call the method addScoops as such:

```
RobIceCream.addScoops(1);
```

If you want to enquire how many scoops Mark has on his ice cream. Your inclination is to do something like this:

```
alert("Mark has " + MarkIceCream.numOfScoops + " scoops");
```

However, MarkIceCream.numOfScoops is not valid because numOfScoops is a private variable and not a public variable.

So the correct way of doing above is:

```
alert("Mark has " + MarkIceCream.getScoops() + " scoops");
```

because getScoops() is a public method and it returns the number of scoops.

So Mark ate his ice cream up:

```
MarkIceCream.consume();
```

To see how many scoops Mark has now:

```
alert("Mark has " + MarkIceCream.getScoops() + " scoops");
```

This should result in the following

```
Mark has 0 scoops.
```

I included this question here because it is extremely important that you understand objects, instances, methods, variables, private and public. I have left out static or class variables and methods but if you are interested someday we can talk more. This should suffice your level of understanding.

For the next five questions, determine if the message valid for the given method definition. Indicate clearly that the message is valid or not valid. Assume the following:

- these variables exist and have been properly initialized,

```
var i=4;
var d=5;
var o = new Test();
```

- the partial method definitions or declarations listed on the right are in the **constructor** of the Test object.

a. o.setData(22, d);   this.setData = function(val1, val2){…}

[✓ VALID]   INVALID

From above on the left hand side, we expect a public method call setData with 2 parameters.

b. o.question = 5;   var question;

VALID   [✓ INVALID]

From above on the left hand side, we expect question to be a public variable ie. declared as this.question = something. However, we have var question; means that question is declared as private variable.

c. o.setMin(i)   this.setMin = function(val1, val2){…}

VALID   [✓ INVALID]

We expect setMin to be public and have one parameter, Here we have 2 paramaters needed ie. va1l and val2. Therefore the function setMin is technically INVALID because val2 is not defined.

d. o.level = 6;   this.level = function(val1) {…}

VALID   [✓ INVALID]

We expect from the left hand side level to be a public variable. It is not a function/method because it is no written as level(). But the right hand side level is declared as a method/function. Therefore, it is not consistent and therefore INVALID.

e. o.setLevel(2)   this.setLevel(newLevel) = function() {…}

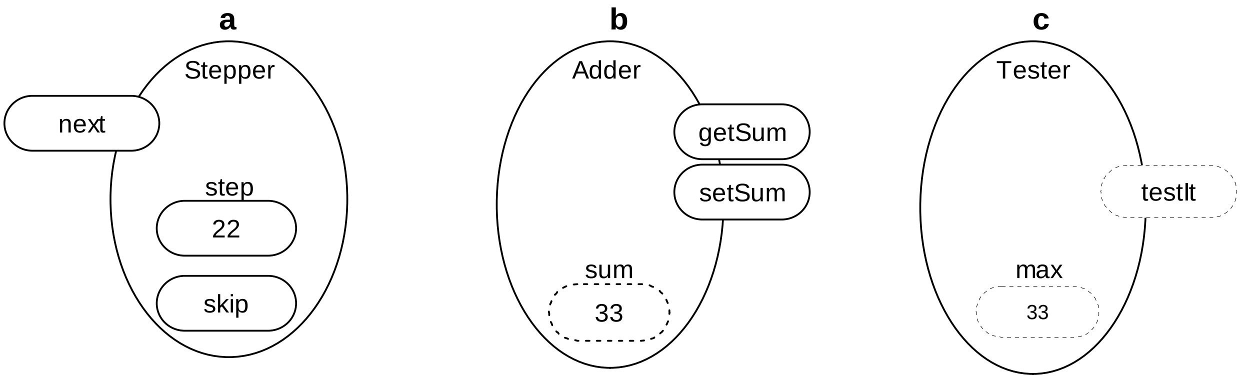
VALID   [✓ INVALID]

From looking at the LHS, you expect setLevel to be public with one parameter. So you expect the following declaration:

this.setLevel = function(newLevel) { ….. }

The RHS has an incorrect definition of the method and therefore an INVALID answer.

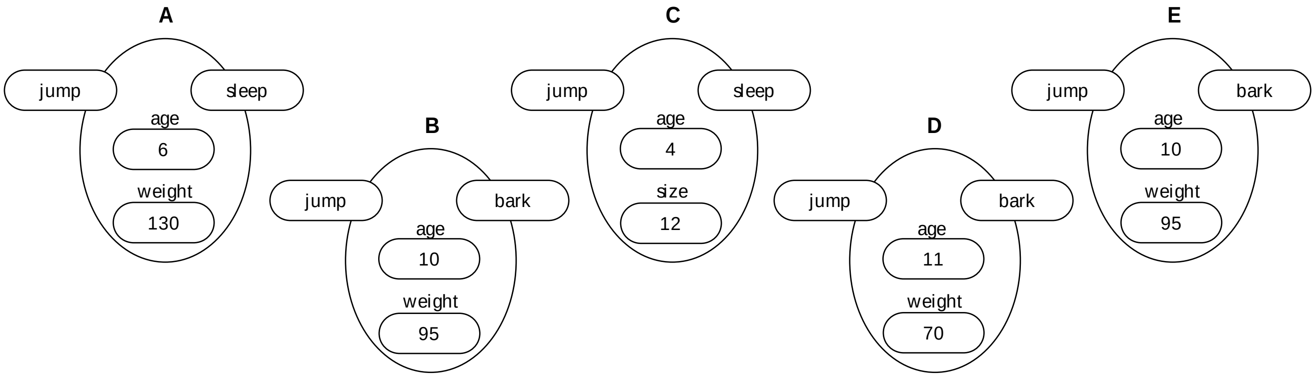
Consider the following object diagrams for three different classes of objects:



Which of these diagrams are valid object diagrams?

**A.	a only**
B.	b only
C.	a and b only
D.	b and c only
E.	none

Consider the following five complete object diagrams:



Which one of the following is a valid statement about these diagrams?

A.	Only B and E could be instances of the same class.
B.	Only A and C could be instances of the same class.
**C.	Only B, D, and E could be instances of the same class.**
D.	A and C could be instances from one class, and B, D and E from another class.
E.	All five could be instances of the same class.

Consider the following complete implementation of the Location class:

```
      function Location (initX, initY){
            var x = initX;
            var y = initY;    // x-y coordinate of the position


            this.setX = function(newX) {
                  x = newX;
            }

            this.setY = function(newY) {
                  y = newY;
            }

            this.getX = function() {
                  return x;
            }

            this.getY = function() {
                  return y;
            }

      }
```

Assume point is a Location object that is defined. Which one of the following code fragments the Exam class will correctly indicate and store that the coordinate of the point object has moved from its current position `(x, y)` to `(x+1, y+1)`?

```
A.    Location (point.getX() + 1, point.getY() + 1);
      // We already declared point as an instance of the class Location.
      // We only want to change the value of x and y NOT create a new object

B.    point.getX() = point.getX() + 1;
      point.getY() = point.getY() + 1;
      //getX() returns the value for x. It is not for changing the value of x
      to a newer value

C.    point.y = point.y + 1;
      point.x = point.x + 1;
      // point.x and point.y is not valid because x and y is not public var.

D.    point.setX(point.x + 1);
      point.setY(point.y + 1);
      // point.x and point.y is not valid because x and y is not public var.
```

`E.    point.setX(point.getX() + 1);`
`      point.setY(point.getY() + 1);`

```
      // The code above is similar to this

      tempX = point.getX();   // get the value of X
      tempY = point.getY();   // get the value of Y

      We want to update X as tempX + 1 and Y as tempY +1
      So you can do this point.setX(tempX + 1) and point.setY(tempY + 1) If
      you replace tempX as point.getX(), you get point.setX(point.getX() + 1)
```

Which one of the following correctly constructs a new Location object based on the constructor definition?

To create an instance variable, you need to have the following:
var instanceName = new className (parameters);

A..

```
Location (11, 22);
// missing “new”
```

B.

```
var start = new location (11, 22);
// location is spelled with small l instead of big L
```

C.

```
var x = 11;
var y = 22;
var new Location (x, y);
// missing the instance name
```

D.

```
var initX = 11;
var initY = 22;
var start = new Location (initX, initY);
// looks good here
```

E.

```
var initX = 11;
var initY = 22;
var start = Location(initX, initY);
// missing new
```

Assume the variables below represent the month and the day of the month, and are assigned values within the valid ranges:

```
var month;  / /  valid range: 1 - 12
var day;    / /  valid range: 1 - 31
```

Which one of the following expresses the interval from 7/22 to 9/22?

A. (month >= 7) && (month <= 9) && (day <= 22)
B. ((month >= 7) && (day >= 22)) || ((month <= 9) && (day <= 22))
C. ((month == 7) || (month = = 9)) && ((day <= 22) || (month == 8))
D. ((month == 7) && (day >= 22)) && ((month == 9) && (day <= 22)) || (month == 8)
E. ((month == 7) && (day >= 22)) || (month == 8) || ((month == 9) && (day <= 22))

A and B is missing month=8 as an alternative.
You need to represent the following

(month equal 7 AND Day greater than or equal 22)
OR (month equal 8)
OR (month equal 9 AND Day less than or equal 22)

Remember AND is represented as &&
OR is represented as ||

So we have

```
(month == 7 && day >= 22)
|| (month == 8)
|| (month == 9 && day <=22)
```

Which of the following boolean expression is equivalent to the following for **all values of A and B**. You can assume that A and B are declared as boolean. Note: if **A** is true, then **!A** will be false.

```
!(A && B);
```

A. !A && !B
B. A && !B
C. A || !B
D. !A || !B
E. None of the above.

To do this you need to create a TRUTH table.

From the question

A	**B**	**A && B**	**! (A && B)**
T	T	T	F
T	F	F	T
F	T	F	T
F	F	F	T

Answer A:

A	**B**	**!A**	**!B**	**!A && !B**
T	T	F	F	F
T	F	F	T	F
F	T	T	F	F
F	F	T	T	T

Answer B:

A	**B**	**!B**	**A && !B**
T	T	F	F
T	F	T	T
F	T	F	F
F	F	T	F

Answer C:

A	**B**	**!B**	**A || !B**
T	T	F	T
T	F	T	T
F	T	F	F
F	F	T	T

Answer D:

A	**B**	**!A**	**!B**	**!A && !B**
T	T	F	F	F
T	F	F	T	T
F	T	T	F	T
F	F	T	T	T

This produces the same result as question

Consider the following poorly indented method in a function named Test:

```
function calculate(x, y, z) {
      if (x == 1)
      if (y == 5)
      return Math.pow (y, x);
      else
      return Math.pow (y, y);
      return Math.pow (y, z);
}
```

Which one of the following values is returned by the message `calculate(1, 2, 3)`?

A. 2

B. 4

C. 5

D. 8

E. none of the above

If we rewrite the above question and properly indent the code we should get

```
function calculate(x, y, z) {
      if (x == 1)
            if (y == 5)
                  return Math.pow (y, x);
            else
                  return Math.pow (y, y);
      return Math.pow (y, z);
}
```

Here x=1, y=2, z=3

So we satisfy (x == 1)
therefore we check whether y==5, in this case it is NOT TRUE so we execute the "else" part and return Math.pow(y,y) and in this case Math.pow(2,2)=4

Consider the following code fragment:

```
      var n = X;
      while (n > 0) {
            alert(n);
            n = n - 1;
      }
      alert(n);


      Testing with X=1, you get the following
      alert(1)
      alert(0)

      Testing with X=0, you get the following
      alert(0)
```

Which one of the following translations of the code above produces the same output as the code given for **all values of X**.

```
A.    for (n = X; n > 0; n++) {
            alert(n);
      }
      alert(n);

Testing with X=1
      alert(1)
      alert(2)
      alert(3) you will see that it is already different so point trying more


B.    var n;
      for (n = X; n > 0; n--) {
            alert(n);
            n = n - 1;
      }
      alert(n);

Testing with X=1
      alert(1) … after that you will execute that n = n-1, so now n=0 and
then in “for loop” n--  means n=n-1 again. This time n = -1. This cause it
terminate the “for loop” because it violates the (n > 0) conditions. Then we
execute the second alert and we get
      alert(-1)
```

```
C.    var n = X;
      do {
            alert(n);
            n = n - 1;
      } while (n > 0);
      alert(n);
```

Testing with X=1

```
      alert(1)
      alert(0)
```

Testing with X=0

```
      alert(0)
      alert(-1)
```

```
D.    var n = X;
      if (n > 0) {
            do {
                  alert(n);
                  n--;
            } while (n > 0);
      }
      alert(n);
```

Testing with X=1

```
      alert(1)
      alert(0)
```

Testing with X=0

```
      alert(0)
```

```
E.    int n = X;
      do {
            if (n > 0)
                  alert(n);
            n--;
      } while (n > 0);
      alert(n);
```

Testing with X=1

```
      alert(1)
      alert(0)
```

Testing with X=0

```
      alert(-1)
```

Consider the following two code fragments:

fragment 1

```
if (a == b) {
      if (c == d) {
            alert("A");
      }
}
else {
      alert("B");
}
```

fragment 2

```
if ((a == b) && (c == d))
      alert("A");
else
      alert("B");
```

Under which of the following conditions will the two fragments produce the same output?

i. Only a and b are the same values.
ii. Only c and d are the same values.
iii. Both a and b are the same values, and both c and d are the same values.

A. i only
B. iii only
C. i and iii only
D. ii and iii only
E. i, ii, and iii

```
Let us consider(a == b) only
LHS (left hand side)
Do nothing

RHS (right hand side)
alert("B")

Let us consider(c == d) only
LHS
alert("B")

RHS
alert("B")


Let us consider BOTH (a == b) and (c == d)
LHS
alert("A")

RHS
alert("A")
```