edhesive
ONLINE LEARNING THAT STICKS

AP COMPUTER SCIENCE A
DIAGNOSTIC EXAM

Multiple Choice Section
Time – 1 hour and 15 minutes
Number of questions – 40
Percent of total grade – 50

Directions: Determine the answer to each of the following questions or incomplete statements, using the available space for any necessary scratch work. Then decide which is the best of the choices given and fill in the corresponding oval on the answer sheet. No credit will be given for anything written on the question sheets. Do not spend too much time on any one problem.

Notes:

- Assume the classes listed in the Quick Reference Guide have been imported where appropriate.
- Assume the declarations of variables and methods appear within the context of an enclosing class.
- Assume that method calls that are not prefixed with an object or class name and are not shown within a complete class definition appear within the context of an enclosing class.
- Unless otherwise noted in the question, assume that parameters in method calls are not null.

1. Consider the following code segment.

```
int x = 20;

while (x < 25)
{
        x++;
        System.out.println(x);
}
```

What are the first and last numbers printed?

	first	last
(A)	20	24
(B)	20	25
(C)	20	26
(D)	21	25
(E)	21	26

2. In order to place a phone call on a mobile phone several factors must be in place. The boolean variables `isOn` and `isRoaming` indicate the state of the phone. The `int bars` stores the numbers of bars from 0 to 5, inclusive.

For a call to be made `isOn` must be true, `isRoaming` must be false, and `bars` must be greater than 0.

Which of the following code segments correctly sets the boolean variable `canCall` to true when it is possible for a call to be made?

I
```
if ( isOn && !isRoaming && bars >0)
            canCall = true;
```

II
```
       if (isOn )
              canCall = true;
       if (!isRoaming)
              canCall = true;
       if (bars >0)
              canCall = true;
```

III
```
   if ( isOn || !isRoaming && bars >0)
              canCall = true;
```

(A) I only
(B) II only
(C) III only
(D) I and II
(E) II and III

3. Assume that an array of integer values has been declared as follows and has been initialized.

```
int list [] = new int[15];
```

Which of the following code correctly swaps the values of list[2] and list[3]?

(A)
```
list[2] = list[3];
list[3] = list[2];
```

(B)
```
int n = list[2];
list[2] = list[3];
list[3] = list[2];
```

(C)
```
int n = 3;
list[3] = list[2];
list[2] = list[n];
```

(D)
```
int n = list[2];
list[2] = list[3];
list[3] = n;
```

(E)
```
int n = list[2];
list[2] = list[3];
list[3] = list[n];
```

4. Consider the following code segment.

```
ArrayList<String> stuff = new ArrayList<String>();
stuff.add("Z");
stuff.add("f");
stuff.add(2, "W");
stuff.remove(1);
stuff.add("x");
System.out.println (stuff);
```

Which is printed as a result of executing this code segment?

(A) `[Z, f, W, x]`
(B) `[W, Z, f, x]`
(C) `[f, W, x]`
(D) `[Z, W, x]`
(E) `[Z, f, x]`

5. When designing a class hierarchy which of the following is NOT true about the parent class?

(A) The parent class should hold any variables and methods common to all of the child classes.
(B) The parent class' variables should be made public so the child class can access them.
(C) It is possible for a parent class to have a parent class itself.
(D) The parent class cannot access any of the methods in the child classes.
(E) The child class will have access to the parent class' public methods and constructor via the super keyword.

Questions 6 and 7 refer to the following code:

```
int x = random number such that 1<= x <=n;
for (int i = 1; i <= x; i++)
  for (int j = 1; j <= x; j++)
      System.out.println ("Yo");
```

6. What is the minimum number of times "Yo" will be printed?

(A) 1
(B) 2
(C) n
(D) n - 1
(E) n2

7. What is the maximum number of times "Yo" will be printed?

(A) 1
(B) 2
(C) n
(D) n - 1
(E) n^2

8. The following method is intended to return the maximum value of the input array.

```
public int findMax (int b[])
     {
          int m = Integer.MIN_VALUE;

/* missing code */

 return m;
     }
```

Which of the code segments shown can be used to replace `/* missing code */` so that `findMax` will work as intended?

I

```
for (int i = 0; i < b.length; i++)
    if ( m > b[i])
    m = b[i];
```

II

```
for (int i = 0; i < b.length; i++)
    if ( m < b[i])
    m = b[i];
```

III

```
for (int i = b.length -1; i >=0; i--)
    if ( m < b[i])
    m = b[i];
```

(A) I only
(B) II only
(C) III only
(D) I and II
(E) II and III

9. Consider the following code segment.

```
String stuff = "VWXYZ";

for (int i = 0; i < stuff.length() - 1; i++)
    System.out.print(stuff.substring( i, i + 1));
```

What is printed as a result of executing this code segment?

(A) `VWXY`
(B) `VWXYZ`
(C) `WXYZ`
(D) `VVVVVWWWWWXXXXXYYYYYZZZZZ`
(E) `VWXYZVWXYZVWXYZVWXYZ`

10. Consider the following method.

```
public void test(int maxNum)
{

int first = 0;
int second = 0;
int third = 0;

for (int k = 1; k <= maxNum; k++)
{
  if (k% 2 == 0 && k % 3 ==0)
      first++;
  if (k % 2 == 0)
      second++;
  if (k % 3 == 0)
      third++;
        }
        System.out.println(first + " " + second + " " + third);
}
```

What is printed as a result of the call `test (30)`?

(A) `5 15 5`
(B) `5 15 10`
(C) `5 10 5`
(D) `5 10 10`
(E) `30 15 10`

11. Consider the following method that is intended to remove all appointments before noon from an ArrayList of appointments. The `Time` class stores all time in military time, so AM appointments are between 0 and 11:59, PM are 12:00 onward.

```
public void removeAm (ArrayList <Time> ap)
{
  for (int i =0; i < ap.size(); i++)
  {
     if( /* code here */)
        ap.remove(i);
  }

}
```

Which of the following could be used to replace `/* code here */` so `remove` will work as intended?

(A) `ap[i].getHour() < 12`
(B) `ap.get(i).getHour() < 12`
(C) `ap.get(i).getHour().compareTo(2) < 0`
(D) `ap.get(i) < 12`
(E) `ap.get(i).getHour().remove(12)`

12. Consider the following incomplete method:

```
public int doSomething (int val)
     {
       /* missing code */
     }
```

The table below lists input and output values for this method.
For example, `doSomething(4)` returns `3`, `doSomething(5)` returns `4`, etc.

Input	Output
4	3
5	4
12	6
15	14
18	9
20	19
21	20
24	12

Which of the code segments shown can be used to replace `/* missing code */` so that `doSomething` will work as intended?

I

```
if (val % 2 ==0 && val % 3 == 0)
    return val / 2;
return val - 1;
```

II

```
if (val % 2 ==0 && val % 3 == 0)
    return val / 2;
else
    return val - 1;
```

III

```
if (val % 2 ==0 )
    if ( val % 3 == 0)
    return val /2;
else
    return val -1;
```

(A) I only
(B) II only
(C) III only
(D) I and II
(E) I, II and III

13. Consider the following method:

```
public void mystery(int x)
{
  if ( x > 1)
       mystery (x / 10);
  System.out.println(x);
}
```

What would be output by mystery(54321) ?

(A)
```
0
```

B)
```
54321
4321
321
1
0
```

(C)
```
54321
5432
543
54
5
0
```

(D)
```
0
5
54
543
5432
54321
```

(E)
```
0
1
21
321
4321
54321
```

14. Consider the following instance variables and incomplete method as a part of a class that represents an appointment. The variables startHour and startMinute are used to represent the start time of the appointment, and duration represents the length of the appointment in minutes.

```
    private int startHour;
    private int startMinute;
    private int duration;

public void pushAppointment(int min)
    {
      /* code missing */
    }
```

Which of the following code segments could be used to replace `/*code missing*/` so that the method works as intended?

I

```
int m = startMinute + min;
int h = startHour * 60 + m;

startMinute = m % 60;
startHour = h / 60 %12;
if(startHour == 0) startHour = 12;
```

II

```
startMinute = startMinute + min;
startHour = (startHour + (startMinute /60)) % 60;
startMinute = startMinute % 60;
if(startHour == 0) startHour = 12;
```

III

```
startMinute = startMinute + min;
startHour = (startHour + (startMinute /60)) % 12;
startMinute = startMinute % 60;
if(startHour == 0) startHour = 12;
```

(A) I only
(B) II only
(C) III only
(D) I and III only
(E) I,II And III

15. Consider the following method

```
public String  inSchoolMessage(int age)
{
if( age < 5 || age > 18)
return "not school aged";
else
return "school aged";
}
```

Consider the following code segments that could be used to replace the body of inSchoolMessage.

I.
```
if( age >= 5)
{
 if (age <= 18)
return "school aged";
else
return "not school aged";
}
else
return "not school aged";
```

II.
```
if( age < 5)
        return "not school aged";
else if (age > 18)
        return "not school aged";
else
        return "school aged";
```

III.
```
if( age >= 5)
return "school aged";
else if (age <= 18)
return "school aged";
else
return "not school aged";
```

(A) I only
(B) II only
(C) III only
(D) I and II only
(E) I,II And III

16. Consider the following class.

```
import java.lang.*;

public class WhatsIt

{
  private int value;

  public WhatsIt(int v)
  {
   value = Math.abs(v%60);
  }

  public void increment ()
  {
   value = (value + 1) %60;
  }

  public int getVal ()
  {
   return value;
  }
}
```

The following code appears in another class:

```
WhatsIt w1 = new WhatsIt ( 76);
WhatsIt w2 = new WhatsIt ( 59);
WhatsIt w3 = w2;
w2.increment();
w2.increment();
w2.increment();
System.out.println(w1.getVal() + " " + w2.getVal() + " " + w3.getVal());
```

What is output?

(A) `16 61 59`
(B) `76 2 2`
(C) `16 2 2`
(D) `16 2 59`
(E `16 61 61`

17. The following incomplete method is intended to reverse the input array parameter.

```
public void reverse (int a[])
  {
     int temp [] = new int [a.length];

     for (int i =0; i < a.length; i++)
        temp [i] = a[ /*missing code */];

     for (int i =0; i < a.length; i++)
        a[i] = temp[i];

  }
```

Which of the following could be used to replace /*missing code*/ so that executing the code would reverse the array.

(A) i
(B) a.length -i -1
(C) a.length -1
(D) a.length - i
(E) a.length + 1

18. Assume that x and y are boolean variables and have been properly initialized.

```
(x || y) && !(x || y)
```

The result of evaluating the expression above is best described as

(A) Always true
(B) Always false
(C) true only when x is true and y is true
(D) true only when x and y have the same value
(E) true only when x and y have different values

19. Assume the following variable declarations have been made:

```
double r = Math.random();
int c;
```

Which of the following assigns a value to `c` that is uniformly distributed between `-10 <= c < 10`?

(A) `(int)(r * (-10) - 20)`
(B) `(int)(r * 21 - 11)`
(C) `r * 21 - 11`
(D) `(int)r * 21 - 11`
(E) `(int)(r * 20 - 10)`

20. Consider the following method `isIncreasing` which is intended to return true if each element in the array is greater than the element before it, otherwise it should return false.

```
public boolean isIncreasing (int a[])
     {
          boolean increasing = /* expression */;

          for (int i=1; i < a.length; i++)
            {
               /* loop body */
               }
          return increasing;
     }
```

/* expression */ /* loop body */

(A) false

```
if (a[i-1] > a[i])
   increasing = false;
```

(B) true

```
if (a[i-1] > a[i])
   increasing = false;
```

(C) true

```
if (a[i-1] > a[i])
   increasing = false;
else
   increasing = true;
```

(D) false

```
if (a[i-1] > a[i])
   increasing = false;
else
   increasing = true;
```

(E) false

```
if (a[i-1] > a[i])
   increasing = false;
```

21. What is stored in the array?

```
int a [] [] = new int [3][2];

for (int r = 0; r < a.length; r++)
{
   for(int c = 0; c < a[r].length; c++)
   {
     a[r][c] = r + c;
    }
}
```

(A)

0	1	2
1	2	3
2	3	4

(B)

0	1
1	2
2	3

(C)

2	3
3	4
4	5

(D)

0	1	2
1	2	3

(E)

1	2	3
2	3	4

22. What is stored in the following array?

```
for (int r = 0; r < a.length; r++)
{
    for(int c = 0; c < a[r].length; c++)
    {
      a[r][c] = 1;
      if (r % 3 == 0 || c % 3 ==0)
         a[r][c] = 3;
      if ( r % 2 ==0 && c % 2 ==0)
         a[r][c] = 2;
    }
}
```

(A)

1	1	1	1
2	2	2	2
3	3	3	3
1	1	1	1

(B)

2	2	2	2
2	1	2	3
2	2	2	2
2	3	2	3

(C)

2	2	2	2
2	1	2	1
2	2	2	2
2	1	2	3

(D)

2	1	2	3
1	1	1	1
2	1	2	1
3	1	1	3

(E)

2	3	2	3
3	1	1	3
2	1	2	3
3	3	3	3

23. Consider the following code:

```
String s = "ABCDEFG";
```

What is output by:

```
System.out.println(s.substring(s.length()/2, s.length()));
```

(A) `ABCD`
(B) `ABCDEFG`
(C) `DEFG`
(D) `DEF`
(E) `EFG`

24. Consider the following code:

```
int x = 7 + 4;
double y = x / 2;
```

What is output by:

```
System.out.println(y);
```

(A) `4.0`
(B) `4.5`
(C) `5.0`
(D) `5.5`
(E) `Error, incompatible types`

25. Consider the following code:

```
public void doStuff (String s)
{
   System.out.println(s.substring(0, 1));
}
   public void doStuff (double i)
{
   System.out.println(i / 10);
}
```

What is output by:

```
doStuff (543);
```

(A) `4`
(B) `5`
(C) `54`
(D) `54.3`
(E) `None of the above`

26. Consider the following code:

```
int x = 95;

    while (x > 50)
    {
      System.out.print(x % 7 + " ");
      if (x % 2 == 0)
         x -= 10;
      else
         x--;
     }
```

What is output by:

```
System.out.println(s.substring(s.length()/2, s.length()));
```

(A) `4 1 5 2 6`
(B) `5 2 6 3 0`
(C) `4 1 5 2 6`
(D) `4 3 0 4 1 5`
(E) `4 3 2 1 0 6 5 4 3 2 1 0 6 5 4 3 2 1 0 6 5 4 3 2 1 0 6 5 4 3 2 1 0 6 5 4 3 2 1 0 6 5 4 3 2`

27. Consider the following method:

```
public String mystery (String s, String sub)
{
  if (s.indexOf(sub) >= 0)
     return s;
  return sub + s.substring(0, sub.length());

}
```

What is output as a result of `System.out.println(mystery("computer science", "java"));` ?

(A) `computer science`
(B) `comsci`
(C) `javacomp`
(D) `scicom`
(E) `compjava`

28. Consider the following code segment:

```
int a = /* value */;
int b = /* value */;

boolean t = ( a >= b);

t = (a != b) && t;
```

Which of the following best describes the conditions under which `t` is true after the code is executed?

(A) When a < b
(B) When a > b
(C) When a == b
(D) t is always true
(E) t is always false

29. Consider the following code segment:

```
for (int i = 1; i < n; i++)
      {
         for (int j = 0; j < n; j++)
         {
            System.out.print ( i + j + " " );

         }

      }
```

What is output when `n = 3`?

(A) `1 2`
(B) `1 2 3 1 2 3`
(C) `2 3 4 2 3 4`
(D) `1 2 3 2 3 4`
(E) `1 2 3 4 2 3 4 5 3 4 5 6`

30. Consider the following class declarations.

```
public class Appliance
{
	private boolean on;

	public Appliance ()
	{
		on = true;
	}
	public Appliance (boolean o)
	{
	on = o;
	}
	}

public class Toaster extends Appliance
{
	public Toaster (boolean t)
	{
	super (t);
	}
}
```

Which of the following statements will NOT compile?

(A) `Toaster t1 = new Toaster (false);`
(B) `Toaster t2 = new Toaster ();`
(C) `Appliance t3 = new Toaster(true);`
(D) `Appliance t4 = new Appliance ();`
(E) `Appliance t = new Appliance (false);`

31. Assume that `x` and `y` are `int` values. The expression

```
!((x >= y) || !(x < y))
```

Evaluates to which of the following?

(A) Always true
(B) Always false
(C) True when x is equal to y
(D) True only when x is greater than y
(E) True only when x is less than y

32. Consider the following code segment:

```
int x = 4;
int y = 142;
while (x <= y)
{
  int m = Math.abs( x - y);
  y = y / x;

  System.out.print(m + " ");
}
```

What is output as a result of executing the code segment?

(A) 2
(B) 4
(C) 138 31 4
(D) 35 8 2
(E) 138

33. Consider the following method:

```
public int doStuff (int x)
{
   int val =0;
   for ( int i = 2; i < x; i+=2)
   {
      for (int j = 2; j < x; j+=2)
      {
          val = val +  i + j;
      }
   }

   return val;
 }
```

What value is returned by the call `doStuff(5)`?

(A) 16
(B) 18
(C) 24
(D) 32
(E) 72

34. Consider the following incomplete method howMany

```
      //returns the count of how many times the int val is in the array a
      //returns 0 if not in the array

  public int howMany (int a[], int val)
  {
    int c =0;

    /* missing code */

    return c;
  }
```

For example consider the following code segment

```
  int a [] = {49, 28 , 36 , 21 , 26 , 28, 61 , 22 , 11 , 28 , 13, 37};

  System.out.println(howMany(a, 28));
```

This should result in the value 3 being printed. Which of the following could be used to replace /* missing code */ so that howMany will work as intended?

(A)

```
for (int i =0; i < a.length; i++)
    {
       if (val == a[i])
       {
          c++;
          break;
       }
    }
```

(B)

```
for (int i =0; i < a.length; i++)
    {
        if (val == a[i])
           return c;
        }
```

(C)

```
 for (int i =0; i < a.length; i++)
     {
         if (val == a[i])
            return i;
         }
```

(D)

```
for (int i =0; i < a.length; i++)
     {
       if (val == a[i])
            c++;
         }
```

E)

```
for (int i =0; i < a.length; i++)
     {
        if (val != a[i])
             c++;
          }
```

35. Consider the following code segments:

I.
```
for (int i = 4; i <= 32;i += 4)
{
  System.out.print(i + " ");
}
```

II.
```
int x = 0;
while(x < 32 )
{
    x += 4;
    System.out.print(x + " ");
}
```

III.
```
for (int i = 4; i <= 32;i ++)
{
    System.out.print(i%4 + " ");
}
```

Which correctly prints the numbers `4,8,12, ..., 32`?

(A) I only
(B) II only
(C) III only
(D) I and II only
(E) I, II and III

36. Consider the following method.

```
public void alterIt (int a[])
{

   for (int i =0; i < a.length; i++)
   a[i] = a[i] %  2;

}
```

What would the following code print.

```
   int list [] = { 45, 56,78, 34, 92, 23, 18};

alterIt(list);

for (int i = 0; i < list.length; i++)
  System.out.print( list [i] + " " );
```

(A) 0 1 0 1 0 1 0

(B) 45 56 78 34 92 23 18

(C) 1 0 0 0 0 1 0

(D) 1 0 1 0 1 0 1

(E) 0 1 1 1 1 0 1

37. Consider the following declaration of the class `Ascending` that is intended to add random numbers to an `ArrayList` of `Integers` in ascending order.

```
public class Ascending
{
  private ArrayList <Integer> list;


  public Ascending ( )
  {
     // code missing
  }

I

ArrayList <Integer> m  = new ArrayList <Integer> ();

m.add(0, new Integer ( (int)(Math.random() * 20)));
for (int i = 1; i < 20; i++)
    m.add(i, new Integer (list.get(i-1) + (int)(Math.random() * 20)));

list = m;
```

II

```
ArrayList <Integer> list  = new ArrayList <Integer> ();

list.add(0, new Integer ( (int)(Math.random() * 20)));
for (int i = 1; i < 20; i++)
    list.add(i, new Integer (list.get(i-1) + (int)(Math.random() * 20)));
```

III

```
list  = new ArrayList <Integer> ();

list.add(0, new Integer ( (int)(Math.random() * 20)));
for (int i = 1; i < 20; i++)
  list.add(i, new Integer (list.get(i-1) + (int)(Math.random() * 20)));
```

(A) I only
(B) II only
(C) III only
(D) I & II
(E) I, II & III

38. Consider the following code segment.

```
double a = 18.4;
double b = .78;

if ( b * (a - 5) != (a * b + b * -5))
   System.out.println("error");
```

III

```
list  = new ArrayList <Integer> ();

list.add(0, new Integer ( (int)(Math.random() * 20)));
for (int i = 1; i < 20; i++)
    ist.add(i, new Integer (list.get(i-1) + (int)(Math.random() * 20)));
```

Which of the following best describes why `"error"` would be printed ?
Remember that mathematically `b(a - 5) = b*a + b*(-5)`.

(A) Numeric Cast error
(B) Modular Division
(C) Roundoff error
(D) Overflow
(E) Incorrect negative signs

39. Consider the following recursive method.

```
public String recur(int n, int b)
{
  String oct = " " + n % b;
  if (n / b > 0)
    return  recur (n/b, b) + oct;

  return oct;
}
```

What is executed as a results of executing the following statement?

```
System.out.println(recur (66, 5));
```

(A) 1 1
(B) 2 3
(C) 2
(D) 1 3 2
(E) 2 3 1

40 Consider the following method:

```
public static String repeat (String word, int a)
{
  if (a > word.length())
     return word;
  return word + repeat (word.substring(0, word.length() - 1), a);
}
```

What is printed as a result of executing the following statement?

```
System.out.println (repeat("gopher", 5));
```

(A) gopher
(B) gophergophegophgopgog
(C) ggogopgophgophegopher
(D) gophergophegoph
(E) gophgophegopher