A+ Computer Science
M/C Written Test

General Directions:

1) DO NOT OPEN EXAM UNTIL TOLD TO DO SO.

2) **NO CALCULATORS of any kind may be used.**

3) You have 45 minutes to complete this contest. If you are in the process of actually writing an answer when the signal to stop is given, you may finish writing that answer.

4) Papers may not be turned in until forty-five minutes have elapsed. If you finish the test before the end of the allotted time, remain at your seat and retain your paper until told to do otherwise. You may use this time to check your answers.

5) All answers must be written on the answer sheet/Scantron card provided. Indicate your answers in the appropriate blanks provided on the answer sheet or on the Scantron card. Clean erasures are necessary for accurate Scantron grading.

6) You may place as many notations as you desire anywhere on the test paper except on the answer sheet or Scantron card which is reserved for answers only.

7) You may use additional scratch paper provided by the contest director.

8) All questions have ONE and only ONE correct (BEST) answer. There is a penalty for all incorrect answers. **All provided code segments are intended to be syntactically correct, unless otherwise stated (i.e. `error` is an answer choice). Ignore any typographical errors and assume any undefined variables are defined as used.**

9) A reference to commonly used Java classes is provided with the test and you may use this reference during the contest. You may detach the reference sheets from the test booklet but DO NOT DO SO UNTIL THE CONTEST BEGINS.

10) Assume that any necessary import statements for Standard Java 12 Packages and classes (e.g. .lang, .util, System, Math, Double, etc.) are included in any programs or code segments that refer to methods from these classes and/or packages.

Scoring:

1) All questions will receive 6 points if answered correctly; no points will be given or subtracted if unanswered; 2 points will be deducted for each incorrect answer.

Standard Classes and Interfaces — Supplemental Reference

class java.lang.Object
- boolean equals(Object other)
- String toString()
- int hashCode()

interface java.lang.Comparable<T>
- int compareTo(T other)
  Return value < 0 if this is less than other.
  Return value = 0 if this is equal to other.
  Return value > 0 if this is greater than other.

class java.lang.Integer implements Comparable<Integer>
- Integer(int value)
- int intValue()
- boolean equals(Object obj)
- String toString()
- int compareTo(Integer anotherInteger)
- static int parseInt(String s)

class java.lang.Double implements Comparable<Double>
- Double(double value)
- double doubleValue()
- boolean equals(Object obj)
- String toString()
- int compareTo(Double anotherDouble)
- static double parseDouble(String s)

class java.lang.String implements Comparable<String>
- int compareTo(String anotherString)
- boolean equals(Object obj)
- int length()
- String substring(int begin, int end)
  Returns the substring starting at index begin and ending at index (end - 1).
- String substring(int begin)
  Returns substring(from, length()).
- int indexOf(String str)
  Returns the index within this string of the first occurrence of str. Returns -1 if str is not found.
- int indexOf(String str, int fromIndex)
  Returns the index within this string of the first occurrence of str, starting the search at the specified index.. Returns -1 if str is not found.
- charAt(int index)
- int indexOf(int ch)
- int indexOf(int ch, int fromIndex)
- String toLowerCase()
- String toUpperCase()
- String[] split(String regex)
- boolean matches(String regex)

class java.lang.Character
- static boolean isDigit(char ch)
- static boolean isLetter(char ch)
- static boolean isLetterOrDigit(char ch)
- static boolean isLowerCase(char ch)
- static boolean isUpperCase(char ch)
- static char toUpperCase(char ch)
- static char toLowerCase(char ch)

class java.lang.Math
- static int abs(int a)
- static double abs(double a)
- static double pow(double base, double exponent)
- static double sqrt(double a)
- static double ceil(double a)
- static double floor(double a)
- static double min(double a, double b)
- static double max(double a, double b)
- static int min(int a, in b)
- static int max(int a, int b)
- static long round(double a)
- static double random()
  Returns a double value with a positive sign, greater than or equal to 0.0 and less than 1.0.

interface java.util.List<E>
- boolean add(E e)
- int size()
- Iterator<E> iterator()
- ListIterator<E> listIterator()
- E get(int index)
- E set(int index, E e)
  Replaces the element at index with the object e.
- void add(int index, E e)
  Inserts the object e at position index, sliding elements at position index and higher to the right (adds 1 to their indices) and adjusts size.
- E remove(int index)
  Removes element from position index, sliding elements at position (index + 1) and higher to the left (subtracts 1 from their indices) and adjusts size.

class java.util.ArrayList<E> implements List<E>

class java.util.LinkedList<E> implements List<E>, Queue<E>

Methods in addition to the List methods:
- void addFirst(E e)
- void addLast(E e)
- E getFirst()
- E getLast()
- E removeFirst()
- E removeLast()

class java.util.Stack<E>
- boolean isEmpty()
- E peek()
- E pop()
- E push(E item)

interface java.util.Queue<E>
- boolean add(E e)
- boolean isEmpty()
- E peek()
- E remove()

class java.util.PriorityQueue<E>
- boolean add(E e)
- boolean isEmpty()
- E peek()
- E remove()

interface java.util.Set<E>
- boolean add(E e)
- boolean contains(Object obj)
- boolean remove(Object obj)
- int size()
- Iterator<E> iterator()
- boolean addAll(Collection<? extends E> c)
- boolean removeAll(Collection<?> c)
- boolean retainAll(Collection<?> c)

class java.util.HashSet<E> implements Set<E>

class java.util.TreeSet<E> implements Set<E>

interface java.util.Map<K,V>
- Object put(K key, V value)
- V get(Object key)
- boolean containsKey(Object key)
- int size()
- Set<K> keySet()
- Set<Map.Entry<K, V>> entrySet()

class java.util.HashMap<K,V> implements Map<K,V>

class java.util.TreeMap<K,V> implements Map<K,V>

interface java.util.Map.Entry<K,V>
- K getKey()
- V getValue()
- V setValue(V value)

interface java.util.Iterator<E>
- boolean hasNext()
- E next()
- void remove()

interface java.util.ListIterator<E> extends java.util.Iterator<E>

Methods in addition to the `Iterator` methods:
- void add(E e)
- void set(E e)

class java.lang.Exception
- Exception()
- Exception(String message)

class java.util.Scanner
- Scanner(InputStream source)
- boolean hasNext()
- boolean hasNextInt()
- boolean hasNextDouble()
- String next()
- int nextInt()
- double nextDouble()
- String nextLine()
- Scanner useDelimiter(String pattern)

A+ Computer Science Contest #2425-04 October 26, 2024

Note: Correct responses are based on **Java SE Development Kit 20 (JDK 20)** from Oracle, Inc. All provided code segments are intended to be syntactically correct, unless otherwise stated (e.g., "error" is an answer choice) and any necessary Java SE 20 Standard Packages have been imported. Ignore any typographical errors and assume any undefined variables are defined as used. **For all output statements, assume that the System class has been statically imported using: import static java.lang.System.***

QUESTION 1

Which of the following is equivalent to the number 212_{10}?

A. 421_{7} B. 111_{14} C. 256_{9} D. 314_{8} E. 183_{11}

QUESTION 2

What is output by the code to the right?

A. 151 B. 125
C. 189 D. 137
E. There is no output due to an error.

```
out.println(4 * 32 - 5 + 12 / 5);
```

QUESTION 3

What is output by the code to the right?

A. 45.45 B. 45.46
C. 4560.45 D. 45.4560
E. There is no output due to an error.

```
out.printf("%4.2f",45.456);
```

QUESTION 4

What is output by the code to the right?

A. elloTherel
B. elloThereo
C. elloThereoelloThereo
D. elloTherelelloTherel
E. www.apluscompsci.com

```
String s = "www.apluscompsci.com";
s = "HelloThere";
s = s.substring(1) + s.charAt(3);
s.concat(s);
out.println(s);
```

QUESTION 5

What is output by the code to the right?

A. True B. False C. false D. true
E. There is no output due to a syntax error.

```
boolean t = true;
boolean f = false;
boolean r = t & f | f;
r |= t ^ f;
out.println(r);
```

QUESTION 6

What is output by the code to the right?

A. 6 B. 7 C. 7.0 D. 6.0
E. There is no output due to an error.

```
double g = 6.7;
int a = Math.round(g);
out.println(a);
```

QUESTION 7

What is the output by the code to the right?

A. 123 B. 13
C. 23 D. 12
E. There is no output due to an error.

```
int g = 3;
if(g++ < 8)
  out.print(1);
if(++g <= 4)
  out.print(2);
if(g-- > 4)
  out.print(3);
```

QUESTION 8

What is the output by the code to the right?

A. 80
B. 83
C. 86
D. 84
E. There is no output due to an error.

```
out.println('a' - 'F' + '8');
```

QUESTION 9

What is output by the code to the right?

A. 24
B. 30
C. 28
D. 33
E. There is no output due to an error.

```
int[] i = new int[6];
for(int j = 0; j < 6; j++)
  i[j] = j * j;
i[3] += i[4] += i[2];
out.println(i[3] + i[1]);
```

QUESTION 10

What is output by the code to the right?

A. 25
B. 30
C. 22
D. 33
E. There is no output due to an error.

```
int sum = 0;
for(int y = 0; y < 5; y++)
  for(int x = y; x < 7; x++)
     sum++;
out.println(sum);
```

QUESTION 11

What is the output by the code to the right?

A. 7
B. 9
C. 8
D. There is no output due to a compile error.
E. There is no output due to a runtime error.

```
String s = "HAPPY HALLOWEEN!";
s = "ABC DEFG HIJKL MNOP";
s+="QRS TUV WX Y Z123 456 7890";
Scanner sc = new Scanner(s);
sc.next();
sc.next();
sc.next();
s = sc.next();
sc.next();
s += sc.next();
out.println(s.length());
```

QUESTION 12

What is the output by the code to the right?

A. 11
B. 59
C. -2
D. 50
E. There is no output due to an error.

```
out.println(45 ^ 32 - 12 | 11);
```

QUESTION 13

What is the correct order of precedence for the operators to the right ?

A. II, III, IV, I
B. I, II, III, IV
C. III, IV, I, II
D. I, III, II, IV
E. III, II, IV, I

```
I. -> (lambda expression)
II. !=
III. instanceof
IV. ?: (ternary)
```

QUESTION 14

What is the output by the code to the right ?

A. 4
B. 32
C. 64
D. 8
E. 16

```
out.println(Integer.SIZE);
```

QUESTION 15

What is the worst-case runtime of finding and removing an item from a doubly linked list?

A. O(NlogN) B. O(logN) C. O(N) D. O(1) E. O(N^2)

QUESTION 16

What is output by the code to the right?

A. `227` B. `242`

C. `237` D. `231`

E. There is no output due to an error.

```
String c = "practice.apluscompsci.com";
c = "Ap2";
int sum = 0;
for(char ch:c.toCharArray())
  sum += ch;
out.println(sum);
```

QUESTION 17

What is output by the line marked `//q17`?

A. `5` B. `4`

C. `7` D. `3`

E. There is no output due to an error.

QUESTION 18

What is output by the line marked `//q18` ?

A. `413975`
B. `579134`
C. `975`
D. `41310864`
E. There is no output due to an error.

```
ArrayList<Integer> a;
a = new ArrayList<Integer>();
for(int y=4;y<11;y++)
 a.add(y);
a.removeIf(y -> y % 2 == 0);
out.println(a.size()); //q17
a.add(13);
a.add(4);
Collections.reverse(a);
a.forEach(y -> out.print(y)); //q18
```

QUESTION 19

What is output by the code to the right?

A. `11` B. `5`

C. `9` D. `6`

E. There is no output due to an error.

```
int[] ij = new int[6];
for(int i = 0; i < 5; i++)
  ij[i+1] = ++ij[i];
for(int j = 0; j < 5; j++)
  ij[j+1]++;
out.println(ij[5]);
```

QUESTION 20

If an insertion sort algorithm (assume it is operating at average case runtime) sorts a list of 1,000 items in 3 seconds, how long will the same algorithm take to sort a list of 4,000 items?

A. 48 seconds B. 24 seconds C. 64 seconds D. 16 seconds E. 32 seconds

QUESTION 21

What is output by the code to the right?

A. `to6` B. `too`

C. `toh` D. `Pablitoo`

E. There is no output due to a runtime error.

```
String h = "Pablito";
h += h.offsetByCodePoints(2, 4);
h = h.substring(5);
out.println(h);
```

QUESTION 22

What is output by the line marked `//q22` in the code to the right?

A. `14` B. `26` C. `21`
D. `18` E. `28`

QUESTION 23

What is output by the line marked `//q23` in the code to the right?

A. `78` B. `64` C. `59`
D. `54` E. `83`

QUESTION 24

What is output by the line marked `//q24` in the code to the right?

A. `212` B. `63` C. `78`
D. `87` E. `73`

```
public int recur(int y) {
if(y <= 0)
 return 0;
if(y % 5 == 2)
 return 4 + recur(y - 3);
if(y % 3 == 1)
 return 7 + recur(y - 2);
return 2 + recur(y - 1) + recur(y / 2);
}

////////////client code///////////////
out.println(recur(8)); //q22
out.println(recur(11)); //q23
out.println(recur(14)); //q24
```

QUESTION 25

What is output by the line marked `//q25` in the code to the right?

A. `true false`
B. `false false`
C. `false true`
D. `true true`
E. There is no output due to a runtime error.

QUESTION 26

What is output by the line marked `//q26` in the code to the right?

A. `true false`
B. `false false`
C. `false true`
D. `true true`
E. There is no output due to a runtime error.

```
String s1 = "Abracadabra";
String s2 = "Hocus Pocus";
String r = "[A-W]\\w+.\\w+";
String o = "" + s1.matches(r);
o += " " + s2.matches(r);
out.println(o); //q25

r = "(\\w\\D\\S){3}.{2,}";
o = "" + s1.matches(r);
o += " " + s2.matches(r);
out.println(o); //q26
```

QUESTION 27

What is output by `//q27` in the code to the right?

A. `1` B. `3`
C. `2` D. `null`
E. There is no output due to a runtime error.

QUESTION 28

What is output by `//q28` the code to the right?

A. `7` B. `9`
C. `6` D. `5`
E. There is no output due to a runtime error.

```
Map m = new TreeMap<>();
m.put(3,7);
m.put(11,9);
m.put(11,5);

out.println(m.size()); //q27

out.println(m.put(3,6)); //q28
```

QUESTION 29

Which of the following is not a legal instantiation?

A. `Object a2 = new TreeSet<Integer>();`
B. `Map a3 = new TreeMap<String,String>();`
C. `Float a4 = new Float(9.4);`
D. `Collection a1 = new LinkedList<Integer>();`
E. All of the above are legal instantiations.

QUESTION 30

What could replace **<1*>** in the code to the right so that the `Wolf` class must define the `bite` method?

A. `implement int bite()`
B. `abstract int bite()`
C. `int bite()`
D. `private int bite()`
E. `interface int bite()`

QUESTION 31

What could replace **<2*>** in the code to the right so that the `Wolf` class private field `noise` is set to the string `"awoo"`?

A. `noise = "awoo"`
B. `super("awoo")`
C. `super() ;`
`noise = "awoo"`
D. A or C
E. All of the above.

QUESTION 32

What is output by the line marked `//q32` code to the right?

A. `72`
B. `64`
C. `68`
D. `36`
E. There is no output due to an error

QUESTION 33

What is output by the line marked `//q33` code to the right?

A. `awooawoo`
B. `Woofawoo`
C. `awooWoof`
D. `WoofWoof`
E. There is no output due to an error.

```
abstract class Canine{

int teeth;
private String noise;

public Canine() {
 noise = "Woof";
 teeth = 32;
}

public String howl() {
 return noise;
}

<1*>;

}

class Wolf extends Canine{

private String noise;

public Wolf() {
 <2*>;
 teeth = 36;
}

public int bite() {
 return teeth;
}

}

//////////////////////////////////////
Canine c = new Wolf();
Wolf w = new Wolf();
int q32 = c.bite()+w.bite();
String q33 = c.howl()+w.howl();
out.println(q32); //q32
out.println(q33); //q33
```

QUESTION 34

Which could replace **<1*>** in the code to the right so that the `flip` method works as intended?

A. `b.push(a.pop())`

B. `b.push(a.peek())`

C. `a.pop()`

D. A and C.

E. All of the above.

QUESTION 35

Which of the following could replace **<2*>** in the code to the right so that the `get` method works as intended?

A. `0`

B. `a.size()`

C. `b.size()`

D. B and C.

E. All of the above.

QUESTION 36

Assuming **<1*>** and **<2*>** are filled correctly, what is output by the line marked `//36` in the client code to the right?

A. `-2`

B. `212`

C. `321`

D. `17`

E. There is no output due to an error.

QUESTION 37

Assuming **<1*>** and **<2*>** are filled correctly, what is output by the line marked `//37` in the client code to the right?

A. `[104, -25, 4, 17, 212, 9]`

B. `[104, -25, 4, -2, 17, 212]`

C. `[212, 17, -2, 4, 2, 104]`

D. `[9, 212, 17, 4, -25, 104]`

E. There is no output due to an error.

QUESTION 38

What kind of data structure is represented by the `Structure` class to the right?

A. Linked List

B. Stack

C. Binary Search Tree

D. Queue

E. Priority Queue

```
class Structure<E>{
 Stack<E> a, b;
 public Structure() {
  a = new Stack<E>();
  b = new Stack<E>();
 }
 public void add(E e) {
  a.add(e);
 }
 public E get() {
  return get(<2*>);
 }
 public E get(int i) {
  flip(i + 1);
  E e = b.peek();
  flop(b.size());
  return e;
 }
 public E remove() {
  return remove(a.size());
 }
 public E remove(int i) {
  flip(i + 1);
  E e = b.pop();
  flop(b.size());
  return e;
 }
 private void flip(int i) {
  while(i-- > 0 && !a.isEmpty())
    <1*>;
 }
 private void flop(int i) {
  while(i-- > 0 && !b.isEmpty())
    a.push(b.pop());
 }
 public String toString() {
  flip(a.size());
  String n = ""+b;
  flop(b.size());
  return n;
 }
}
//////////////////////////////////////
Structure<Integer> list;
list = new Structure<Integer>();
list.add(9);
list.add(212);
list.add(17);
list.add(-2);
list.remove();
list.add(321);
out.println(list.get(2)); //q36
list.add(4);
list.add(-25);
list.remove(2);
list.add(104);
out.println(list); //q37
```

QUESTION 39

What is the height of the binary tree created by inserting the following numbers in the given order?

```
34 56 78 22 16 9 36 37 29 54 7 25 14 212 43
```

QUESTION 40

What is the post-order traversal of the binary tree created for question 39?