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 #2425-01 October 05, 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 37_{10}?

A. 211_4 B. 100101_2 C. 25_{16} D. 45_8 E. All are equivalent.

QUESTION 2

What is output by the code to the right?

A. 7 B. 24
C. 6 D. 19
E. There is no output due to an error.

```
out.println(3 * 8 + 2 / 4);
```

QUESTION 3

What is output by the code to the right?

A. 25|1B|true B. 25|1B|11010100
C. 25|1b|true D. 25|1b|false
E. There is no output due to an error.

```
out.printf("%o|%X|%b",21,27,212);
```

QUESTION 4

What is output by the code to the right?

A. 2 > 7
B. 2 > 77
C. 212 > >
D. 212 > 7>
E. There is no output due to an error.

```
String s="212 > 7";
s=s.substring(2);
s+=s.charAt(4);
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 a=true;
boolean b=false;
a &= a || b;
out.println(a ^ b);
```

QUESTION 6

What is output by the code to the right?

A. 2.0 B. 17.0 C. -17.0 D. -2.0
E. There is no output due to an error.

```
int y=17;
double d=Math.copySign(y, -2);
out.println(d);
```

QUESTION 7

What is the output by the code to the right?

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

```
if(true || false)
  out.print(1);
else
  out.print(2);
  out.print(3);
```

QUESTION 8

What is the output by the code to the right?

A. `HelloWorld`
B. `Hello`
C. `Hello['W', 'o', 'r', 'l', 'd']`
D. Output cannot be determined until runtime.
E. There is no output due to an error.

```
String f="Hello";
char[] c= {'W','o','r','l','d'};
out.println(f+" "+c);
```

QUESTION 9

What is output by the code to the right?

A. `0`
B. `-4`
C. `2`
D. `-1`
E. There is no output due to an error.

```
int a = 5;
do{
  a = a - 3;
}while( a > 0 );
out.println( a );
```

QUESTION 10

What is output by the code to the right?

A. `[4, 6, 12, 5, 6, 12, 6]`
B. `[4, 6, 4, 5, 6, 12, 0]`
C. `[4, 3, 4, 5, 6, 12, 0]`
D. `[4, 6, 4, 0, 6, 5, 0]`
E. There is no output due to an error.

```
int y[]= {2,3,4,5,6,1,0};
y[y[y[3]]]=y[y[2]];
y[5]=y[y[y[3]]]+y[4];
y[y[6]]=y[y[y[6]]];
out.print(Arrays.toString(y));
```

QUESTION 11

What is the output by the code to the right?

A. `213`
B. `149`
C. `199`
D. `187`
E. There is no output due to an error.

```
String t;
t="23 51 46 78 9 64 37 84 29 14 dun";
Scanner s = new Scanner(t);
int sum=0;
while(s.hasNextInt()){
     if(s.nextInt()%2==1)
       sum+=s.nextInt();
}
out.print(sum);
```

QUESTION 12

What is the output by the code to the right?

A. `149`
B. `31`
C. `61`
D. `23`
E. There is no output due to a compile error.

```
out.println(19 ^ 15 >> 2 << 3 | 21);
```

QUESTION 13

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

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

```
I. *
II. |(bitwise)
III. |(logical)
IV. &(logical)
```

QUESTION 14

What is the output by the code to the right?

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

```
int[] sizes=new int[4];
sizes[0]=Integer.SIZE;
sizes[1]=Float.SIZE;
sizes[2]=Character.SIZE;
sizes[3]=Short.SIZE;
Arrays.sort(sizes);
out.println(sizes[1]);
```

QUESTION 15

What is the runtime of adding an item to a stack?

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

QUESTION 16

Which of the following can replace **<*1>** in the code to the right so that the code segment compiles without error?

I. `26.2`
II. `new Float(26.2)`
III. `new Float("26.2f")`

A. I only
B. II only
C. III only
D. I and II only
E. II and III only

```
ArrayList<Float> decs;
decs = new ArrayList<Float>();
decs.add(<1*>);
```

QUESTION 17

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

A. `6`
B. `$`
C. `&`
D. `4`
E. There is no output due to an error.

QUESTION 18

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

A. `6`
B. `5`
C. `4`
D. `W`
E. There is no output due to an error.

```
ArrayList<Character> cList;
cList = new ArrayList<Character>();
cList.add('$'); cList.add('&');
cList.add('7'); cList.add('4');
cList.add('W');
ListIterator iter;
iter = cList.listIterator();
iter.next(); iter.next();
iter.remove();
iter.add('6');
iter.previous();
out.println(iter.next());       //q17
iter.set('5');
iter.next();
iter.previous(); iter.previous();
out.println(iter.next());       //q18
```

QUESTION 19

What could replace **<1*>** in the code to the right, so that the code will compile?

A. `static`
B. `default`
C. `protected`
D. A and B.
E. All of the above.

```
interface I{
  public int ret();
  <1*> void show(int y) {
     out.println(y*y);
  }
}
```

QUESTION 20

Which of the following reserved words is used to access a member of a parent class?

A. `extends` B. `implements` C. `parent` D. `super` E. `above`

QUESTION 21

Which sorting algorithm is implemented in the code to the right?

A. Selection sort
B. Insertion sort
C. Quick sort
D. Merge sort
E. Tim Sort

```
int[]k= {2,5,1,-9,0,5,3,9};
Arrays.sort(k);
String h=Arrays.toString(k);
out.println(h);
```

QUESTION 22

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

A. `6`
B. `5`
C. `7`
D. `3`
E. `4`

QUESTION 23

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

A. `11`
B. `8`
C. `10`
D. `12`
E. `9`

QUESTION 24

What is the purpose of method `recur` to the right?

A. Count all the even integers in the given string.
B. Count all the consonants in the given string.
C. Count all the vowels in the given string.
D. Count all the characters with even ASCII values in the given string.
E. Count all the capital letters in the given string.

```
public int recur(String h) {
 if(h.length() < 1)return 0;
 if(h.charAt(0) % 2 == 0)
  return 1 + recur(h.substring(1));
 return 0 + recur(h.substring(1));
}
////////////client code///////////////
out.println(recur("212 is")); //22
out.println(
         recur("Greater than 7")); //23
```

QUESTION 25

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

A. `true`
B. `[0, 17, ABC]`
C. `false`
D. `[212, g]`
E. No output due to an error.

QUESTION 26

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

A. `[0, 17, ABC]`
B. `[33, 9, USA]`
C. `[212, g]`
D. `[0, 17, 212, 33, 9, ABC, USA, g]`
E. No output due to an error.

```
TreeSet<String>a,b;
a=new TreeSet<String>();
b=new TreeSet<String>();
a.add(""+0);
a.add(""+212);
a.add(""+17);
a.add("ABC");
a.add("g");
b.add("33");
b.add("USA");
b.add("212");
b.add("g");
b.add("9");
out.println(a.retainAll(b));  //q25
out.print(a);  //q26
```

QUESTION 27

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

A. `true false`
B. `false false`
C. `false true`
D. `true true`
E. No output due to an error.

QUESTION 28

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

A. `true true`
B. `true false`
C. `false true`
D. `false false`
E. No output due to an error.

```
String a = "We run the numbers";
String b = "Like Kevin(Malone)";
String s1 = "\\W+";
String s2 = "[a-zA-z ]{4,}";
String o = ""+a.matches(s1);
o += " "+b.matches(s1);
out.println(o); //q27
o = ""+a.matches(s2);
o += " "+b.matches(s2);
out.println(o); //q28
```

QUESTION 29

What is output by the code to the right?

A. `6142` B. `3552`

C. `4663` D. `4972`

E. No output due to an error.

```
int i=0x239 + 05432 + 0b10001101;
out.println(i);
```

QUESTION 30

What is output by the code to the right?

A. `0`

B. `2147483647`

C. `-2147483648`

D. `2147483648`

E. No output due to an error.

```
out.print(Integer.MAX_VALUE+1);
```

QUESTION 31

Which of the following does not match the regex pattern to the right?

A. `""`

B. `"w1ww1ww1ww1ww1wd2d2"`

C. `"e3d33333333"`

D. `"w1ww1wa7"`

E. All of the above match the pattern.

```
(\\w\\d\\w)*([^aeiou]\\d)*
```

QUESTION 32

What could replace **<1*>** in the code to the right so that `rows` is equal to the number of rows in `mat`?

A. `mat.length()`

B. `mat.length`

C. `mat.rows`

D. A and B.

E. All of the above.

QUESTION 33

What is output by the client code to the right?

A. `24` B. `29`

C. `23` D. `28`

E. There is no output due to an error.

```
public static int lights(int[][] mat){
  int tot = 0;
  int rows = <1*>;
  int cols = mat[0].length;
  int start = Math.max(rows, cols) - 1;
  int m = start / 2;
  for(int i = start; i >= 0; i--){
    tot += mat[i][m];
    tot += mat[m][i];
  }
  return tot;
}
//////////////client code/////////////
int[][]mat=new int[3][3];
mat[0]=new int[]{2, 4, 5};
mat[1]=new int[]{6, 5, 7};
mat[2]=new int[]{3, 1, 9};
out.println(lights(mat));
```

QUESTION 34

Given the following measurements, what is the most likely running time for method `sample(int[] data)` where N is equal to `data.length`? Choose the most restrictive correct answer.

Value of N	Time for method `sample` to complete
2,000	1 second
4,000	4 seconds
8,000	16 seconds

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

QUESTION 35

Which of the following could replace **<1*>** in the code to the right so that the `B` class initializes properly and sets the `i` instance variable to the `i` parameter?

A. `super.i=i;`

B. `super(i);`

C.
```
super();
super.i=i;
```

D. A and C.

E. All of the above.

QUESTION 36

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

A. `5`

B. `Not`

C. `Not:5`

D. Output cannot be determined until runtime.

E. There is no output due to an error.

QUESTION 37

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

A. `Not:12`

B. `Not:5`

C. `12`

D. Output cannot be determined until runtime.

E. There is no output due to an error.

QUESTION 38

Assuming **<1*>** is filled correctly, what is output by the line marked `//38` in the client code to the right?

A. `Not:5`

B. `Not:8`

C. `8`

D. Output cannot be determined until runtime.

E. There is no output due to an error.

```
class A
{
 int i;
 public A()
 {
  i=5;
 }
}
class B extends A
{
 String s;
 public B(int i)
 {
  <1*>
  s="Not";
 }
 public String toString()
 {
  return s+":"+i;
 }
}
//////////////client code/////////////
A a = new A();
out.println(a); //q36
B b = new B(12);
out.println(b); //q37
A ab = new B(8);
out.println(ab); //q38
```

QUESTION 39

What is the post-order traversal of the binary tree created when the following values are inserted in order ?

```
17, 212, 9,-3, 75, 43, 90, 2, -76, 32, 71, 18, 16
```

QUESTION 40

What is the last item popped in the following stack pseudocode?

```
push 5
push 6
push 7
pop X
push 4
push 3
pop X
pop X
push 12
pop X
push 9
push 8
push 10
pop X
pop X
push 2
pop X
pop X
push 1
```