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)

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

What is 212_3 plus 2_{12} ?

A. 18_{16} B. 31_8 C. 11010_2 D. 111_4 E. 24_{10}

QUESTION 2

What is output by the code to the right?

A. 0 B. 3 C. 38 D. 42 E. 51

```
out.println( 3 % 2 + 5 * 5 + 12);
```

QUESTION 3

What is output by the code to the right?

A. IHaveNoEnemies

B. IHave
NoEnemies

C. I Have
No Enemies

D. IHaveNo
Enemies

E. IHaveNo
Enemies

```
out.print("I");
out.print("Have");
out.println("No");
out.print(" Enemies");
```

QUESTION 4

What is output by the code to the right?

A. Have Hero'si

B. Have a Hero'sn arc

C. Have a Hero's arc

D. Have aHero'sn arc

E. Have Hero'sin arc

```
String str = "Have a Villain arc";
String word = "Hero's";
String a = str.substring(0,6);
String b = str.substring(13);
out.println(a + word + b);
```

QUESTION 5

What is output by the code to the right?

A. false B. true

```
out.println(false && false || true);
```

QUESTION 6

What is output by the code to the right?

A. 3.0 B. 3.00 C. 2.2 D. 2.20 E. 2.0

```
out.println(Math.ceil(2.12));
```

QUESTION 7

What is output by the code to the right?

A. `3`
B. `3.0`
C. `4`
D. `4.0`
E. `5.0`

```
double a = 3/2;
int b = 2;
double c = .5;
out.println(a / c + b);
```

QUESTION 8

What is output by the code to the right?

A. `a`
B. `d`
C. `abc`
D. `abd`
E. `abcd`

```
int a = 5 + 6;
int b = 5 * 6;
int c = 5 / 6;
if(a == 5.0 + 6.0)
  out.print("a");
if(b == 5.0 * 6.0)
  out.print("b");
if(c == 5.0 / 6.0)
  out.print("c");
else
  out.print("d");
```

QUESTION 9

What is output by the code to the right?

A. `0123456789`
B. `012345678910`
C. `12345678910`
D. `12345678`
E. `123456789`

```
for(int x = 1;x < 10;x++)
{
  out.print(x);
}
```

QUESTION 10

What is output by the code to the right?

A. `77`
B. `29`
C. `25`
D. `20`
E. `17`

```
int[] ar = {8,3,7,2,6,4};
out.println(ar[1] + ar[2] * ar[3]);
```

QUESTION 11

What is output by the code to the right?

A. `4A`
B. `13A`
C. `3BA`
D. `1 3`
E. `4`

```
String str = "1 3 A B 3";
Scanner f = new Scanner(str);
out.println(f.nextInt() + f.next() +
f.next());
```

QUESTION 12

What is output by the code to the right?

A. `42`
B. `47`
C. `45`
D. `62`
E. `17`

```
int sum = 2;
for(int x = 1; x <= 20; x += 4)
{
  sum += x;
}
out.println(sum);
```

QUESTION 13

What is output by the code to the right?

A. `false`
B. `true`
C. `trueup`
D. `falseup`
E. No output due to a runtime error

```
String give = "never";
out.println("never" == give + "up");
```

QUESTION 14

What is output by the code to the right?

A. `213`
B. `-213`
C. `212`
D. `-212`
E. `-211`

```
out.println(~212);
```

QUESTION 15

What is output by the code to the right?

A. `[For, The, Meme]`
B. `[For, The, Dream]`
C. `[For, The, Stream]`
D. `[For, The, Ayeeee Team]`
E. `[Meme, The, For]`

```
ArrayList<String> al;
al = new ArrayList<>();
al.add("Meme");
al.add("Dream");
al.remove(0);
al.add("Stream");
al.remove(0);
al.remove(0);
al.add("Ayeeee Team");
al.add("The");
al.add("For");
Collections.reverse(al);
out.println(al);
```

QUESTION 16

What is output by the code to the right?

A. `[false, false, false, false, false]`
B. `[true, true, true, true, true]`
C. `[true, false, true, true, false]`
D. `[false, true, false, true, false]`
E. `[false, true, false, false, true]`

```
boolean[] ar = new boolean[5];
for(int i = 0; i < ar.length; i++){
  ar[i] = true;
}
for(int x = 1; x < 5; x++)
{
  for(int i = 0; i < 5;i += x)
  {
    ar[i] = ar[i] ? false : true;
  }
}
out.println(Arrays.toString(ar));
```

QUESTION 17

What is output by the code to the right?

A. 0
B. 1
C. 24
D. 60
E. 61

```
int a = 25;
int b = 37;
int c = a & b;
out.println(a ^ b | c);
```

QUESTION 18

What is output by the code to the right?

A. 2
B. 3
C. 4
D. 5
E. 6

```
int cnt = 0;
int a = 5;
int b = 8;
while(a != 0 && b != 0)
{
  if(a > b)
  {
     a %= b;
  }
  else
  {
     b %= a;
  }
  cnt++;
}
out.println(cnt);
```

QUESTION 19

What is output by the code to the right?

A. 1
B. 7
C. 4
D. 6
E. 5

```
int cnt = 2;
String[] s = "Never Give Up Never Back
Down".split(" ");

String vowels = "[aeiu]";
for(int i = 0; i < s.length; i++)
{
  if(s[i].matches(".*" + vowels +".*"))
  {
     cnt++;
  }
  else
  {
     cnt--;
  }
}
out.println(cnt);
```

QUESTION 20

What is returned when method `go(-3)` is called?

A. `0`

B. `-5`

C. `-3`

D. No output due to a runtime error

E. No output due to infinite recursion

QUESTION 21

How many times is the method `go()` called when `go(10)` is called? (The call go(10) itself is counted as one)?

A. `10`

B. `5`

C. `4`

D. `8`

E. `7`

QUESTION 22

How many times is the method `go()` called when `go(21)` is called? (The call go(21) itself is counted as one)?

A. 10

B. `17`

C. `14`

D. `15`

E. `12`

```
public int go(int num)
{
  if(num <= 0)
     return num;
  if(num % 3 == 0)
     return go(num - 2);
  return go(num - 1);
}
```

QUESTION 23

What is output by the code at `//Code 1` only?

A. `-3`

B. `0`

C. `5`

D. `-2`

E. `6`

QUESTION 24

What is output by the code at `//Code 2` only?

A. `[6, 5, 0, -2]`

B. `[-2, 0, 5, 6]`

C. `[-2, 5, 0, 6]`

D. `[6, 0, 5, -2]`

E. Cannot be determined until runtime

```
PriorityQueue<Integer> pq;
pq = new PriorityQueue<>();

pq.add(6);
pq.add(5);
pq.add(-3);
pq.add(-2);
pq.add(0);
out.println(pq.remove()); //Code 1
out.println(pq); //Code 2
```

QUESTION 25

What can properly replace `//Code1` that can properly instantiate the variables `name` and `workEthic` so that they are equal to the input variables `name` and `work`?

A.
```
this.name = name;
this.workEthic = work;
```

B.
```
this.name = name;
workEthic = work;
```

C.
```
name = name;
workEthic = work;
```

D. A and B only

E. A, B, and C

QUESTION 26

What is output by the code at `//Line 1` only?

A. `0`

B. `15`

C. `998`

D. `999`

E. No output due to a syntax error

QUESTION 27

What is output by the code at `//Line 2` only?

A. `0`

B. `15`

C. `998`

D. `999`

E. No output due to a syntax error

QUESTION 28

What is output by the code at `//Line 3` only?

A. `0`

B. `15`

C. `998`

D. `999`

E. No output due to a syntax error

```
class Person {

  String name;
  int workEthic;

  public Person(String name,
                              int work)
  {
     // Code 1
  }

  public String getName()
  {
     return name;
  }

  public int getWorkEthic()
  {
     return workEthic;
  }
}

class HardWorker extends Person
{
  int workEthic;
  public HardWorker(String name)
  {
     super(name,998);
     workEthic = 999;
  }
}

//////////Client Code
Person np = new Person("Normal",15);

Person dg =
    new HardWorker("David Gogghins");

HardWorker kb =
       new HardWorker("Kobe Bryant");

out.println(np.getWorkEthic());
//Line 1

out.println(dg.getWorkEthic());
//Line 2

out.println(kb.getWorkEthic());
//Line 3
```

QUESTION 29

What is output by the code to the right?

A. `[1, 2, 2, 1, 0]`

B. `[4, 3, 2, 1, 0]`

C. `[1, 2, 1, 2, 0]`

D. `[0, 0, 0, 0, 0]`

E. No output due to a runtime error

```
int[] ar = new int[5];
for(int i = 0; i < 4; i++)
{
  for(int j = i; j <= 4 - 1; j++)
  {
     ar[j] = ar[j + 1] + 1;
  }
}
out.println(Arrays.toString(ar));
```

QUESTION 30

What is output by the code to the right?

A. `46` B. `31`

C. `34` D. `114`

E. `43`

```
String str = Integer.toString(34,5);
int number = Integer.parseInt(str,6);
out.println(number);
```

QUESTION 31

What is output by the code to the right?

A. `15`

B. `42`

C. `45`

D. `84`

E. `90`

```
int sum = 0;
for(int i = 1; i <= 15; i++)
{
  for(int j = 0; j < 3; j++)
  {
     sum += j;
  }
}
out.println(sum);
```

QUESTION 32

What is output by the code to the right?

A. `[6, 3, 6, 8, 4, 2, 3]`

B. `[6, 3, 8, 4, 2]`

C. `[8, 4, 6, 3, 2]`

D. `[2, 3, 4, 6, 8]`

E. `[2, 3, 3, 4, 6, 6, 8]`

```
public static ArrayList<Integer>
go(ArrayList<Integer> al)
{
  TreeSet<Integer> ts;
  ts = new TreeSet<>(al);
  return new ArrayList<>(ts);
}
```

QUESTION 33

What is the purpose of the code to the right?

A. To remove all the duplicates and return the arraylist in the same order.

B. To return a clone of the same arraylist.

C. To remove all duplicates from the arraylist and then sort the contents.

D. To sort the arraylist.

E. To return the arraylist in a tree format.

```
//client code
ArrayList<Integer> al;
al = new ArrayList<>();
int[] ar = {6,3,6,8,4,2,3};

for (int i = 0; i < ar.length; i++) {
  al.add(ar[i]);
}
out.println(go(al));
```

QUESTION 34

What is output by the code to the right?

A. `127` B. `128` C. `255` D. `256` E. `8`

```
out.println(Byte.MAX_VALUE);
```

QUESTION 35

What could correctly fill out `Code 1` so that `Sort` sorts the array in ascending order?

A.
```
int temp = ar[j];
ar[j - 1] = ar[j];
ar[j] = temp;
```

B.
```
int temp = ar[j + 1];
ar[j + 1] = ar[j];
ar[j] = temp;
```

C.
```
int temp = ar[j - 1];
ar[j - 1] = ar[j];
ar[j] = temp;
```

D.
```
int temp = ar[j];
ar[j + 1] = ar[j];
ar[j] = temp;
```

E.
```
ar[j - 1] = ar[j];
ar[j] = ar[j - 1];
```

QUESTION 36

What is output by the code at `Code 2` only?

A. `[4, 12, 23, 36, 56]`

B. `[56, 34, 23, 12, 4]`

C. `[4, 12, 56, 34, 12]`

D. `[12, 4, 56, 23, 34]`

E. `[4, 12, 23, 34, 56]`

QUESTION 37

What type of sort is implemented in the method `Sort`?

A. `Selection Sort`

B. `Bango Sort`

C. `Bubble Sort`

D. `Insertion Sort`

E. `Quick Sort`

QUESTION 38

What is the runtime of the sort to the right?

A. `O(n^2)`

B. `O(n^3)`

C. `O(n)`

D. `O(nlog(n))`

E. `O(2n)`

```
public void Sort(int ar[])
{
  for (int i = (ar.length - 1); i >= 0; i--)
  {
     for (int j = 1; j <= i; j++)
     {
        if (ar[j-1] > ar[j])
        {
           // Code 1
        }
     }
  }
}
```

```
//client code
int[] ar = {12, 4, 56, 23, 34};
Sort(ar);
String str = Arrays.toString(ar);
out.println(str); //Code 2
```

QUESTION 39

Write this equation in prefix notation?

```
3 * 5 + 6 / 2
```

QUESTION 40

Of the 16 possible ordered Quartets, how many will make the expression below true?

```
!(A && B || D) ^ (D ^ !C)
```