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 23 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 #2526-08 January 17, 2026

Note: Correct responses are based on **Java Development Kit 23** 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 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 NOT equivalent to the expression $20313_4 - AB_{12}$?

A. $1B0_{16}$ B. 664_8 C. 12310_4 D. 436_{10} E. 304_{12}

QUESTION 2

What is output by the code to the right?

A. 52 B. 40 C. 72 D. 100

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

```
out.println(98 + 6 && 151 >> 1);
```

QUESTION 3

What is output by the code to the right?

A. %.12990 B. %0.12000\b\b\b99

C. %%.12000\b\b\b99 D. %0.12990

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

```
out.printf("%%%6.5f\\b\\b\\b%d",.12,99);
```

QUESTION 4

How many empty Strings are stored in list by the code to the right?

A. 7 B. 8 C. 0 D. 9 E. 1

```
String k = "AAAaaaaahhhhhhhhh!";
String[] list = k.split("[Aa]");
```

QUESTION 5

How many combinations of a, b, and c will result in a false output by the code to the right?

A. 4 B. 1 C. 0 D. 2 E. 4

```
boolean a = <unknown value>;
boolean b = <unknown value>;
boolean c = <unknown value>;
out.println(a && b || a && c || !a);
```

QUESTION 6

What is output by the code to the right?

A. 125 B. 64 C. 64.0 D. 125.0

E. There is no output due to a runtime error

```
int x = Math.ceilDiv(31, 7);
out.println(Math.pow(x, 3));
```

QUESTION 7

What is output by the code to the right?

A. 26 B. 29 C. 28 D. 27

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

```
int x = 5;
out.println(x+++x+++++x+x++);
```

QUESTION 8

What is output by the code to the right?

A. `STRG`

B. `STRG CRITWEAK`

C. `STRGCRITWEAKFAILHITNR`

D. `STRGCRITWEAK`

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

```
String str ="";
int k = 4;
switch(k)
{
  case 8:
  case 5: str+="POWR"; break;
  case 4: str+="STRG";
  case 7:
  case 6: str+="CRIT";
  case 2: str+="WEAK"; break;
  case 1:  str+="FAIL"; break;
  case 3: str+="HIT";
          str+="NR";
}
out.println(str);
```

QUESTION 9

What is output by the code to the right?

A. `BGLQV[` B. `AFKPUZ` C. `bglqv{` D. `afkpuz`

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

```
for(int i=1; i<=26; i+=5)
  out.print((char)(97+i));
```

QUESTION 10

What is output by the code to the right?

A. `56 80 256 3`

B. `0 0 0 2`

C. `56 80 256 2`

D. There is no output due to a compile error.

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

```
int[] list = new int[0];
list[0] = 56;
list = new int[1];
list[1] = 'P';
list = new int[2];
list[2] = (int)Math.pow(2, 8);
list = new int[3];
list[3] = (int)Math.round(2.499);
for(int k:list)
  out.print(k+" ");
out.println();
```

QUESTION 11

What is output by the code to the right?

A. `-1` B. `220` C. `81` D. `345`

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

```
Scanner input =
  new Scanner("42 81 97 -85 64 35 61 -32 66");
int count = 0;
while(input.nextInt()>0)
  count+=input.nextInt();
out.println(count);
```

QUESTION 12

What is output by the code to the right?

A. `5` B. `91` C. `8` D. `87`

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

```
int[] list = {12, 9, 25, 93, 98, 32, 22};
int[] list2 = {4, 3, 8, 2};
int sum = 0;
for(int i=0; i<list.length; i++)
  sum+=list[i]%list2[(i+7)%list2.length];
out.println(sum);
```

QUESTION 13

What is output by the code to the right?

A. `9` B. `1062` C. `2752` D. `169`

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

```
int x = 3;
int y = 2;
int z = 41;
out.println(z ^ y << x-- * x);
```

QUESTION 14

What could replace **<14*>** such that the output by the code to the right will be 0.

A. `k += Integer.MAX_VALUE;`

B. `k -= Integer.MIN_VALUE/2;`

C. `k -= Integer.MIN_VALUE;`

D. `k += Integer.MAX_VALUE/2;`

E. More than one of these will work

```
int k = Integer.MIN_VALUE;
<14*>
out.println(k);
```

QUESTION 15

Which of the following can replace **<*15>** so the method `use` will output the list?

A. `ArrayList<Character[]>`

B. `ArrayList<>`

C. `ArrayList<Object>`

D. `ArrayList<?>`

E. None of the above will compile

```
public static void use(<*15> list)
{
  for(int i=0; i<list.size(); i++)
     out.println(list.get(i));
}

//CLIENT CODE
ArrayList<String> x =
                    new ArrayList<>();
x.add("stuff");
use(x);
```

QUESTION 16

What is output by the code to the right?

A. `[42, 24.71, 1.1150.9.2, 42.1156.32.3, 45.1153.31.1]`

B. `[45.1153.31.1, 1.1150.9.2, 42.1156.32.3 42, 24.71]`

C. `[1.1150.9.2, 45.1153.31.1, 42.1156.32.3, 24.71, 42]`

D. `[42, 24.71, 45.1153.31.1, 1.1150.9.2, 42.1156.32.3]`

E. `[1.1150.9.2, 24.71, 42, 42.1156.32.3, 45.1153.31.1]`

```
class Dewey implements
Comparable<Dewey>
{
  private String id;
  private int[] sect;
  public Dewey(String x)
  {
     id=x;
     String temp[]=id.split("\\.");
     sect = new int[temp.length];
     for(int i=0; i<temp.length; i++)
        if(!temp[i].isEmpty())
           sect[i] =
           Integer.parseInt(temp[i]);
        else
           sect[i]=0;
  }
  public int compareTo(Dewey k)
  {
     int x = sect.length;
     if(x == k.sect.length)
        return sect[x-1]-k.sect[x-1];
     return x-k.sect.length;
  }
  public String toString()
  {return id;}
}
//CLIENT CODE
ArrayList<Dewey> lib = new
                   ArrayList<Dewey>();
String[] d = {"42.1156.32.3",
              "42",
              "24.71",
              "45.1153.31.1",
              "1.1150.9.2"};
for(String k:d)
{
  lib.add(new Dewey(k));
}
Collections.sort(lib);
out.println(lib);
```

QUESTION 17

Which of the following can replace **<*1>** so that the code to the right would output the following?

```
42.1156.32.3
45.1153.31.1
1.1150.9.2
```

A. `x.contains("115")`

B. `x.indexOf("115") > x.indexOf("\\.")`

C. `x.indexOf(".1") == 2`

D. `x.contains(".11")`

E. `x.sum("\\.") == 3`

```
ArrayList<String> lib = new
                        ArrayList<>();
lib.add("42.1156.32.3");
lib.add("42115");
lib.add("115.71");
lib.add("45.1153.31.1");
lib.add("1.1150.9.2");
lib.stream()
   .filter((x) -> <*1>)
   .forEach((p) -> out.println(p));
```

QUESTION 18

Consider the class from question 16. What is the 2nd line outputted by the code to the right?

A. `9.0.08`

B. `45.1153.31.61`

C. `1.1150.9.2`

D. `42.1156.32.3`

E. There is no output due to a compile error

```
ArrayList<Dewey> lib = new
                       ArrayList<>();
lib.add(new Dewey("42.1156.32.3"));
lib.add(new Dewey("42.115"));
lib.add(new Dewey(".115.71"));
lib.add(new Dewey("45.1153.31.61"));
lib.add(new Dewey("9.0.08"));
lib.add(new Dewey("1.1150.9.2"));
lib.add(new Dewey("8...0"));
lib.stream().filter((x) -> {
  String str = x.toString();
  String[] t = str.split("\\.");
  return t.length > 3;
  }).sorted()
    .forEach((p) -> out.println(p));
```

QUESTION 19

A programmer wants to randomly create a character value between A thru Z and 0 thru 9. Which of the following lines of codes achieves this goal?

A. `char k = (x % 26 < 26) ? (char)('A'+ x % 26) : (char)('0'+(x-26));`

B. `char k = (x % 26 < 10) ? (char)('0'+(x-26)) : (char)('A'+ x % 26);`

C. `char k = (x % 36 < 26) ? (char)('A'+ x % 26) : (char)('0'+(x-26));`

D. `char k = (x % 36 < 10) ? (char)('A'+ x % 26) : (char)('0'+(x-26));`

E. `char k = (x % 36 < 26) ? (char)('0'+(x-26)) : (char)('A'+ x % 26);`

QUESTION 20

What is output by the code to the right?

A. `[naldD, ardHo, ogeScr, Arne, Daffy]`

B. `[Donal, Howar, eScroo, Arne, affy]`

C. `[onald, wardH, oogeSc, Arn, Daff]`

D. `[onald, oward, crooge, rne, affy]c++;`

E. There is no output due to a compile error

```
class Huey
{
  private String k;
  public static int c = 0;
  public Huey(String x)
  {  k=x; c++; }
  public void doStuff()
  {
     String t="";
     if(c < k.length())
     {
        t+=k.substring(c+1);
        t+=k.substring(0,c);
        k = t;
     }
     else
        c=0;
  }
  public String toString()
  {   return k; }
}
//CLIENT CODE
List<Huey> ducks =
   List.of(new Huey("Donald"),
           new Huey("Howard"),
           new Huey("Scrooge"),
           new Huey("Arne"),
           new Huey("Daffy"));
int x = 1;
ducks.forEach(Huey::doStuff);
out.println(ducks);
```

QUESTION 21

What is returned by the method call `myst(15)` based on the code to the right?

A. `151413121110`

B. `10`

C. `15`

D. There is no output due to a Stack Overflow Error

E. There is no output due to a Stack Underflow Error

```
public static int myst(int k)
{
  if(k == 10)
     return k;
  return myst(k+1);
}
```

QUESTION 22

What is returned by the method call `myst(20,10)` based on the code to the right?

A. `44` B. `45` C. `60` D. `61` E. `56`

```
public static int myst(int k, int t)
{
  if(k < 10)
     return 1;
  if(t > 20)
     return 2;
  return t % k + myst(k-1, t+2);
}
```

QUESTION 23

What is returned by the method call

`myst("GORDON","LIGHTFOOT")`

based on the code to the right?

A. `TONFOOTGORDONLIGHTFOOT`

B. `NOTTONFOOTGORDONLIGHTFOOT`

C. `OTONOOTDONTFOOTORDONLIGHTFOOT`

D. There is no output due to a Stack Underflow Error

E. There is no output due to a Stack Overflow Error

```
public static String myst(String k,
                          String t)
{
  if(k.isEmpty())
     return t;
  if(t.isEmpty())
     return k;
  return myst(k.substring(k.length()/2),
              t.substring(t.length()/2))
              + k + t;
}
```

QUESTION 24

What could replace **<24*>** in the code to the right so that the `Connection` class compiles and functions as intended?

A. `abstracts`

B. `implements`

C. `super`

D. `extends`

E. `this`

QUESTION 25

Assume that **<24*>** has been filled in properly. What could replace **<25*>** in the code to the right so that the `Ferry` class compiles and functions as intended?

A.
```
super(a,b);
num = 2;
```

B.
```
new Ferry(a,b);
num = 2;
```

C.
```
num = 2;
super(a,b);
```

D.
```
num = 2;
```

E. More than one of these works

QUESTION 26

Assume that **<24*>** and **<25*>** has been filled in properly. What could replace **<26*>** in the code to the right so that the `Ferry` and **`MountainPath`** classes compiles and functions as intended?

A. `getRoute()`

B. `Connection.getRoute()`

C. `Route.getRoute()`

D. `super.getRoute()`

E. `this.getRoute()`

QUESTION 27

Assume that **<24*>** , **<25*>** , and **<26*>** has been filled in properly. What is the output by the code to the right?

A. `LON-DUNFER2 GEN-MUNMTN PAL-ATHFER2`

B. `LON-EDNCON LON-DUNFER2 DUN-EDNCON GEN-MUNMTN PAL-ATHFER2`

C. `LON-EDN LON-DUN DUN-EDN GEN-MUN PAL-ATH`

D. `LON-EDN LON-DUNFER2 DUN-EDN GEN-MUNMTN PAL-ATHFER2`

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

```
interface Route
{ public String getRoute(); }

class Connection <24*> Route
{
  private String city, city2;
  public Connection(String a,
                    String b)
  {
     city = a;
     city2 = b;
  }
  public String getRoute()
  {
     return city+"-"+city2;
  }
}
class MountainPath extends Connection
{
  public MountainPath(String a,
                      String b)
  {  super(a,b); }
  public String getRoute()
  {
     return <26*>+"MTN";
  }
}
class Ferry extends Connection
{
  private int num;
  public Ferry(String a,
               String b, int n)
  {
     <25*>


  }
  public String getRoute()
  {
     return <26*>
            +"FER"+num;
  }
}
//CLIENT CODE
Route[] list = new Route[5];
list[0] =
       new Connection("LON","EDN");
list[1] =
       new Ferry("LON","DUN",2);
list[2] =
       new Connection("DUN","EDN");
list[3] =
       new MountainPath("GEN","MUN");
list[4] =
       new Ferry("PAL","ATH",1);
for(Route k:list)
{
  out.print(k.getRoute()+" ");
}
```

QUESTION 28

What could replace **<28*>** in the code to the right so the output is false?

A. `"System.System.out.print;"`

B. `"out.println;"`

C. `"System.out.print;"`

D. `"System.out.println;"`

E. more than one of these will result in false.

```
boolean a;
String str;
str = <28*>;
a = str.matches(
    "(System\\.)?out\\.print(ln)?;");
out.println(a);
```

QUESTION 29

What could NOT replace **<29*>** in the code to the right such that the code to the right would behave like a queue?

A. `offer`

B. `addFirst`

C. `add`

D. `addLast`

E. B and D

```
Deque<String> list =
                    new LinkedList<>();
list.<29*>("LON");
list.<29*>("DEN");
list.<29*>("NYC");
list.<29*>("PTL");
list.remove();
list.remove();
list.<29*>("DAL");
list.<29*>("OKC");
list.remove();
out.println(list);
```

QUESTION 30

What could NOT replace **<30*>** in the code to the right such that the code to the right would behave like a stack?

A. `addFirst`

B. `push`

C. `add`

D. `addLast`

E. C and D

QUESTION 31

Assume that **<30*>** has been filled in properly such that the code to the right behaves like a stack. What is the output by the code to the right?

A. `[DAL, DEN, LON]`

B. `[PTL, DAL, OKC]`

C. `[DEN, LON, DAL]`

D. `[DAL, LON, PTL]`

E. `[OKC, DAL, PTL]`

```
Deque<String> list =
                    new LinkedList<>();
list.<30*>("LON");
list.<30*>("DEN");
list.<30*>("NYC");
list.<30*>("PTL");
list.pop();
list.pop();
list.<30*>("DAL");
list.<30*>("OKC");
list.pop();
out.println(list);
```

QUESTION 32

What is output by the code to the right?

A. `92 74 92 74 92 74`

B. `92 4 74 85 60 69 29 18`

C. `92 4 85 84 71 29 18`

D. `92 4 74 85 74 85 74 85`

E. There is no output due to a runtime error

```
int[] nums = {74,85,84,71,80};
int[] vals = {92,4,20,60,69,29,18};
System.arraycopy(nums,1,vals,2,3);
for(int k:vals)
  out.print(k+" ");
```

QUESTION 33

What could replace **<33*>** in the code to the right so that values are add to the map in the code to the right?

A. `push`

B. `add`

C. `offer`

D. `put`

E. more than one of these will work

QUESTION 34

Assuming **<33*>** has been filled in properly, what could replace **<34*>** in the code to the right such that it prints all the correct values held in the map?

A. `map`

B. `map.keySet()`

C. `keySet()`

D. `map.valueSet()`

E. more than one of these will work

QUESTION 35

Assuming **<33*>** and **<34*>** has been filled in properly, what is the output by the line **<35*>** for the code on the right?

A. `19 45 63 72`

B. `72 45 19 63`

C. `37 40 66 93`

D. `72 45 63 19`

E. `37 40 93 66`

QUESTION 36

Assuming **<33*>** and **<34*>** has been filled in properly, what is the output by the line **<36*>** for the code on the right?

A. `{19=66, 37=72, 40=45, 45=40, 63=93, 66=19, 72=37, 93=63}`

B. `{19=66, 45=40, 63=93, 72=37}`

C. `{37=72, 40=45, 66=19, 72=37, 93=63}`

D. `{37=72, 40=45, 66=19, 93=63}`

E. There is no output due to a runtime error

```
TreeMap<Integer,Integer> map;
map = new TreeMap<Integer,Integer>();
map.<33*>(37, 72);
map.<33*>(40, 45);
map.<33*>(93, 63);
map.<33*>(66, 19);
for(Integer k: <34*>)
  out.print(map.get(k)+" ");
out.println(); //<35*>
for(Integer k: map.keySet())
  map.<33*>(map.get(k),k);
out.println(map); //<36*>
```

QUESTION 37

Which of the following is not a legal Java statement?

A. `Set _$ = new HashSet();`

B. `Map _ = new TreeMap();`

C. `Collection $7 = new LinkedList();`

D. `ArrayList $_ = new ArrayList();`

E. More than one of these are not legal statements

QUESTION 38

What is output by the code to the right?

A. 415 B. 225 C. 452 D. 192

E. The output is an infinite loop

```
int sum = 0;
for(int k = 99; k > 72; k-=8)
  for(int t = k; t > 0; t/=3)
     sum += k/t;
out.println(sum);
```

QUESTION 39

What is the 8 bit two's compliment representation of the following decimal number?

121_{10}

QUESTION 40

What is the height of the binary search tree created by inserting the following values in order from left to right?

11 24 79 29 35 25 9 49 21 95 97 10 59 83 43