# 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 #2526-02 October 25, 2025

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 equivalent to the expression $101_2 + 101_2$?

A. $B_{16}$ B. $8_{10}$ C. $1010_2$ D. $1100_2$ E. All are equivalent.

**QUESTION 2**

What is output by the code to the right?

A. 27 B. 25 C. 17
D. 20
E. There is no output due to an error.

```
out.println(212 / 3 - 50);
```

**QUESTION 3**

What is output by the code to the right(&'s indicate blank spaces)?

A. &&&nope
B. &&&NOPE
C. nope&&&
D. NOPE&&&
E. There is no output due to an error.

```
String s = "nope";
out.printf("%7S",s);
```

**QUESTION 4**

What is output by the code to the right?

A. CDEFGBE
B. CDEFG135
C. ABBE
D. There is no output due to a compile error.
E. There is no output due to a runtime error.

```
String s = "", f = "";
s += "ABCDEFG";
f = s.charAt(1) + s.charAt(4);
s = s.substring(2);
out.println(s + f);
```

**QUESTION 5**

What is output by the code to the right?

A. true B. false

```
boolean a = false;
boolean b = true;
a = b & a ^ (b |!a & b);
out.println(b);
```

**QUESTION 6**

What is output by the code to the right?

A. 5 B. 5.0 C. 6.0 D. 6
E. There is no output due to a compile error.

```
double a = 5.555;
int i = Math.round(a);
out.println(i);
```

**QUESTION 7**

What is the output by the code to the right?

A. 20 B. 10
C. 0 D. 120
E. There is no output due to a compile error.

```
int i = 1, j = i;
if(++i > j)
  out.print(1);
if(j-- == i)
  out.print(2);
out.print("0");
```

**QUESTION 8**

What is the output by the code to the right?

A. `35 16 8` B. `36 8 9`

C. `44 16 9` D. `45 8 10`

E. There is no output due to an infinite loop.

```
int i = 0, j = 128, k = 1;
while(i < j) {
  i += k;
  j -= i;
  k++;
}
out.println(i+" "+j+" "+k);
```

**QUESTION 9**

What is output by the code to the right?

A. `26` B. `22`

C. `24` D. `20`

E. There is no output due to an error.

```
int a = 10;
int b = 8 | 16 + a;
int c = b - 8 ^ 4;
out.println(c);
```

**QUESTION 10**

What is the output by the code to the right ?

A. `65 94` B. `60 96`

C. `68 100` D. `70 102`

E. There is no output due to an error.

```
int[] arr = new int[] {
  12, 34, 32, 8, 87};
for(int i = 0; i < 4; i++) {
  arr[i] += arr[i + 1]--;
}
String s = arr[1]+" "+arr[3];
out.println(s);
```

**QUESTION 11**

Which of the following import statements is required so that the code on the right compiles without error?

A. `import java.util.*;`
B. `import java.text.*;`
C. `import java.io.*;`
D. A and B.
E. A and C.

```
public void run()throws Exception{
  File f = new File("text.dat");
  Scanner file = new Scanner(f);
  int times = file.nextInt();
  file.nextLine();
  while(times-- > 0) {
     //implementation
  }
}
```

**QUESTION 12**

What is the output by the code to the right?

A. `200` B. `220`
C. `196` D. `212`
E. There is no output due to an error.

```
String s = "aB1";
int sum = 0;
for(char c:s.toCharArray())
  sum += c;
out.println(sum);
```

**QUESTION 13**

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

A. `I, IV, III, II` B. `IV, I, II, III`

C. `I, IV, II, III` D. `IV, I, III, II`

E. `I, II, III, IV`

```
I. && (logical)
II. =
III. ?: (ternary)
IV. | (bitwise)
```

**QUESTION 14**

What is the output by the code to the right ?

A. `832` B. `864`

C. `1632` D. `1664`

E. `3232`

```
int[] i = new int[5];
i[0] = Double.SIZE;
i[1] = Short.SIZE;
i[2] = Byte.SIZE;
i[3] = Integer.SIZE;
i[4] = Float.SIZE;
Arrays.sort(i);
out.println(i[1]+""+i[4]);
```

**QUESTION 15**

What is the output by the line marked `//q15` in the code to the right?

A. `[0, 9, 8, 7]` B. `[0, 9, 8, 7, 6]`

C. `[2, 1, 0, 9]` D. `[2, 1, 0, 9, 8]`

E. There is no output due to an error.

**QUESTION 16**

What is the output by the line marked `//q16` in the code to the right?

A. `16` B. `14` C. `17` D. `18`

E. There is no output due to an error.

**QUESTION 17**

What is the output by the method call concluded by the line marked `//q17` in the code to the right?

A. `2107110221071102`

B. `0221071102210711`

C. `1022107110221071`

D. `0121071100221071102`

E. There is no output due to an error.

```
ArrayList<Integer> a;
a = new ArrayList<Integer>();
for(int y = 0; y < 10; y++)
  a.add(y);
Collections.rotate(a, -4);
Collections.reverse(a);
out.println(a.subList(3,7)); //q15
a.addAll(a);
a.removeIf(n -> (n % 4 == 1 &&
  a.indexOf(n) != a.lastIndexOf(n)));
out.println(a.size()); //q16
Collections.rotate(a, 6);
a.forEach(n -> {
  if(n % 2 == 1)
     out.print(0);
  if(n % 5 == 3)
     out.print(2);
  if(n == 7)
     out.print(n);
  if(n / 2 > 2)
     out.print(1);
}); //q17
```

**QUESTION 18**

What is the output by the line marked `//q18` code to the right?

A. `false true` B. `true true`

C. `true false` D. `false false`

E. There is no output due to an error.

**QUESTION 19**

What is the output by the line marked `//q19` code to the right?

A. `false true` B. `true true`

C. `true false` D. `false false`

E. There is no output due to an error.

```
String s1 = "Hello Darkness";
String s2 = " My old friend.";
String m1 = "(\\w+) (\\S+)";
String m2 = "(\\s?\\w+.?)+";
String res = "" + s1.matches(m1);
res += " " + s2.matches(m1);
out.println(res); //q18
res = "" + s1.matches(m2);
res += " " + s2.matches(m2);
out.println(res); //q19
```

**QUESTION 20**

Which of the following data structures has the best worst-case runtime for inserting an object at the end?

A. Array B. Hashtable

C. Singly-Linked List D. Binary Search Tree

E. More than one of the above has the best runtime.

**QUESTION 21**

What is the output by the code to the right?

A. `17` B. `21`

C. `36` D. `31`

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

```
int sum = 1;
for(int i = 0; i < 5; i++)
  for(int j = i; j < i * 2; j++)
     for(int k = 0; k < 3; k++)
       sum++;
out.println(sum);
```

**QUESTION 22**

Which of the following could replace **`<1*>`** in the code to the right so that the `swim` method of the `Mako` class returns twice the value the `swim` method of the `Shark` class returns?

A. `super.swim(2)`  B. `super.swim() * 2`

C. `super(2)`  D. `swim() * 2`

E. More than one of the above.

**QUESTION 23**

Which of the following terms describes the scope of the `size`, `speed`, and `name` instance variables in the `Shark` class to the right?

A. `public`  B. `protected`

C. `package protected`  D. `child protected`

E. `private`

**QUESTION 24**

Assuming **`<1*>`** has been filled correctly, what is output by the lines marked `//q24` in the client code to the right?

A. `60 80 40 80`

B. `60 160 40 160`

C. `60 160 80 160`

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

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

**QUESTION 25**

Assuming **`<1*>`** has been filled correctly, what is output by the lines marked `//q25` in the client code to the right?

A. `60 70 60 70`

B. `60 70 60 60`

C. `60 60 60 60`

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

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

**QUESTION 26**

Assuming **`<1*>`** has been filled correctly, what is output by the lines marked `//q26` in the client code to the right?

A. `ChompChompChompChomp`

B. `ChompChompChomp`

C. There is output, followed by a runtime error.

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

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

```
class Shark{
 int size, speed;
 String name;
 private int teeth;
 public Shark
        (int si, int se, String n) {
  name = n;
  size = si;
  speed = se;
  teeth = 60;
  name = n;
 }
 public int swim() {
  return speed;
 }
 public int bite() {
  return teeth;
 }
}
class Mako extends Shark{
 private int teeth;
 public Mako() {
  super(12, 80, "Mako");
  teeth = 70;
 }
 public int swim() {
  return <1*>;
 }
 public void chomp() {
  System.out.print("Chomp");
 }
}
class Hammerhead extends Shark{
 public Hammerhead() {
  super(16, 40, "Hammer");
 }
 public void chomp() {
  System.out.print("ChompChomp");
 }
}
////////////client code////////////
Shark s = new
        Shark(20, 60, "Great White");
Mako m = new Mako();
Hammerhead h = new Hammerhead();
Shark b = new Mako();
String o = "";
o += s.swim()+" "+m.swim();
o += " "+h.swim()+" "+b.swim();
out.println(o); //q24
o = s.bite()+" "+m.bite()+" ";
o += h.bite()+" "+b.bite();
out.println(o); //q25
m.chomp(); //q26
h.chomp(); //q26
b.chomp(); //q26
```

QUESTION 27

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

A. `2@`

B. `22@`

C. `024@`

D. `424@`

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

QUESTION 28

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

A. `4242@`

B. `40142@`

C. `41042@`

D. `0142@`

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

QUESTION 29

What is output by the line marked `//q29` in the client code to the right?

A. `124242@`

B. `424242@`

C. `424142@`

D. `404242@`

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

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

//////////////client code//////////////
out.println(recur(3)); //q27
out.println(recur(8)); //q28
out.println(recur(13)); //q29
```

QUESTION 30

Which of the following could replace **<1*>** in the code to the right so that it can execute without error?

A. `Queue`

B. `List`

C. `LinkedList`

D. A and C.

E. All of the above.

QUESTION 31

Assuming **<1*>** has been filled in correctly, what is output by the code to the right?

A. `Giraffe 1` B. `Elephant 5`

C. `Snake 2` D. `Snail 2`

E. There is no output due to an error.

```
<1*><String> list;
list = new LinkedList<String>();
list.add("Elephant");
list.add("Snake");
list.add("Snail");
list.add("Giraffe");
list.poll();
list.add("Hippo");
list.poll();
list.peek();
out.print(list.poll()+" ");
out.print(list.size());
```

**QUESTION 32**

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

A. `8` B. `3`

C. `11` D. `4`

E. There is no output due to an error.

**QUESTION 33**

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

A. `8` B. `3`

C. `11` D. `4`

E. There is no output due to an error.

```
Stack<Integer> s;
s = new Stack<>();

s.add(8);
s.add(3);
s.add(4);
s.add(7);
s.add(11);


out.println(s.pop()); //q32

s.pop();
s.pop();

out.println(s.pop()); //q33
```

**QUESTION 34**

What is output by the line marked `//q34` code to the right?

A. `6`
B. `-5`
C. `22`
D. `9`
E. `4`

**QUESTION 35**

What is output by the line marked `//q35` code to the right?

A. `22`
B. `6`
C. `-5`
D. `11`
E. `7`

**QUESTION 36**

What is output by the line marked `//q36` code to the right?

A. `0`
B. `22`
C. `11`
D. `-5`
E. `9`

```
int[][] m = {{4,6,7,8,9},
               {11},
              {22,55,0,-5}};


out.println(m[0][0]); //q34


out.println(m[1][0]); //q35


out.println(m[2][2]); //q36
```

QUESTION 37

What is the length of the shortest path in the graph to the right from nodes E to G?

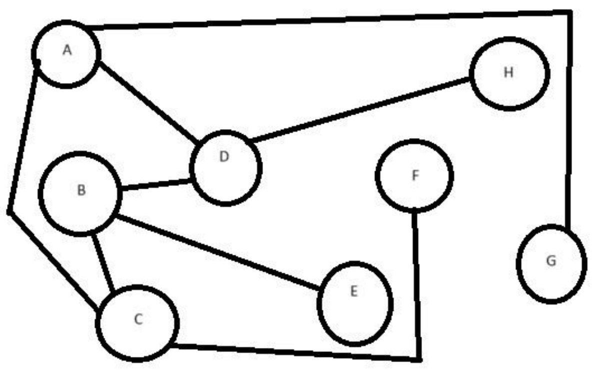
A. 3

B. 2

C. 5

D. 4

E. 6



QUESTION 38

What kind of graph is shown to the right?

A. Connected Unweighted Directed

B. Unweighted Undirected

C. Weighted Undirected

D. Weighted Directed

E. Connected Unweighted Undirected

QUESTION 39

What is the most simplified boolean expression that generates the following truth table?

| A | B | C | Result |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

QUESTION 40

What is the 8-bit two's complement value corresponding to the following number?

$-87_{10}$