**University Interscholastic League**

**Computer Science Competition**

UTCS Invitational - 2010

General Directions (Please read carefully!):

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

2) **No calculator of any kind may be used.**

3) There are 40 questions on this contest exam. 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 45 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. 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, but not on the answer sheet or Scantron card which are 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. Ignore any typographical errors and assume any undefined variables are defined as used.**

9) A reference to commonly used Java classes is provided at the end of the test, and you may use this reference sheet 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 packages and classes (e.g. .util, ArrayList, etc.) are included in any programs or code segments that refer to methods from these classes and 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 an incorrect answer.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Question 1** | | | | |  | | | | | | | E | | | | | | | | | | | |
|  | What does 110112 minus 1112 equal? | | | | | | | | | | | | | | | | | | | | | | |
|  | A. | | 100002 | | | | B. | | | | 101002 | | | | | C. | 111002 | | D. | 1090010 | E. | 102 | |
| **Question 2** | | | | | |  | | | | | | | | B | | | | | double a = 2.5;  double b = a;  a++;  a = a + b + a;  System.out.print(a); | | | | |
|  | | | What is output by the code to the right? | | | | | | | | | | | | | | | |
|  | | | A. | | 7.0 | | B. | | | 7.5 | | | | | C. | 8.5 | | |
|  | | | D. | | 9.5 | | E. | | | 10.5 | | | | | | | | |
| **Question 3** | | | | | |  | | | | | | | | B | | | | | int total = 0;  for(int i = 1; i <= 11; i += 2)  total += 2;  System.out.print(total); | | | | |
|  | | | What is output by the code to the right? | | | | | | | | | | | | | | | |
|  | | | A. | | 2 | | B. | | | 5 | | | | | C. | 6 | | |
|  | | | D. | | 11 | | E. | | | 12 | | | | | | | | |
| **Question 4** | | | | | |  | | | | | | | | B | | | | | String sci = "Floyd";  String fi = "fly";  String last = sci + fi + sci;  last = last.substring(3,6);  System.out.print(last); | | | | |
|  | | | What is output by the code to the right? | | | | | | | | | | | | | | | |
|  | | | A. | | ydflyF | | B. | | | ifi | | | | | C. | fis | | |
|  | | | D. | | oyd | | E. | | | ydf | | | | | | | | |
| **Question 5** | | | | | |  | | | | | | | | B | | | | | int x = 3;  int[] init = {x, x, 2, x + 2, x \* 2};  System.out.print(init.length); | | | | |
|  | | | What is output by the code to the right? | | | | | | | | | | | | | | | |
|  | | | A. | | 5 | | B. | | | 7 | | | | | C. | 11 | | |
|  | | | D. | | 19 | | E. | | | null | | | | | | | | |
| **Question 6** | | | | | |  | | | | | | | | B | | | | | int w = 3;  int z = 2;  int v = w \* -2 - 3 \* z;  System.out.print(v + " " + z); | | | | |
|  | | | What is output by the code to the right? | | | | | | | | | | | | | | | |
|  | | | A. | | 4 3 | | B. | | | -30 2 | | | | | C. | -12 2 | | |
|  | | | D. | | 3 2 | | E. | | | -12 6 | | | | | | | | |
| **Question 7** | | | | | |  | | | | | | | | B | | | | | boolean p = true;  boolean q = false;  boolean r = true;  System.out.print( !(p && q && !r) + " ");  System.out.print( !(r && (p ||q) && !r)); | | | | |
|  | | | What is output by the code to the right? | | | | | | | | | | | | | | | |
|  | | | A. | | true true | | | | B. | | true false | | | | | | | |
|  | | | C. | | false true | | | | D. | | false false | | | | | | | |
|  | | | E. | | true false true | | | | | | | | | | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Question 8** | | |  | | | | | B | | | boolean p = true;  int x = 10;  if( p != false ){  if( x != 5 ){  System.out.print(1);  x = 5;  }  else  System.out.print(2);  if( x == 5 )  System.out.print(3);  }else  System.out.print(4); |
|  | What is output by the code to the right? | | | | | | | | | |
|  | A. | 12 | | B. | | 23 | | | C. | 234 |
|  | D. | 13 | | E. | | 34 | | | | |
| **Question 9** | | |  | | | | | B | | | public class Book{  private final int numPages;  private boolean haveRead;    public Book(int pages){  numPages = pages;  }  public String toString(){  return numPages + " " + haveRead;  }  }  //////////////////////////////////////////  // client code  Book b1 = new Book(100);  System.out.print(b1); // line 1  Book b2 = new Book(100);  System.out.print(b1.equals(b2)); // line 2 |
|  | What is output by the line marked line 1 in the client code to the right? | | | | | | | | | |
|  | A. | 100 false | | | B. | | false | | C. | 0 0 |
|  | D. | 100 true | | | E. | | true | | | |
| **Question 10** | | |  | | | | | B | | |
|  | What is output by the line marked line 2 in the client code to the right? | | | | | | | | | |
|  | A. | true | | B. | | false | | | C. | 0 |
|  | D. | 1 | | E. | | null | | | | |
| **Question 11** | | |  | | | | | B | | | int ax = 40;  int bx = 64;  int cx = ax | bx | bx;  System.out.print(cx + " " + bx); |
|  | What is output by the code to the right? | | | | | | | | | |
|  | A. | 64 64 | | B. | | 0 64 | | | C. | 104 64 |
|  | D. | 127 64 | | E. | | 128 0 | | | | |
| **Question 12** | | |  | | | | | B | | | double st = -3.4;  System.out.print( Math.ceil(st) ); |
|  | What is output by the code to the right? | | | | | | | | | |
|  | A. | -3.0 | | B. | | -2.4 | | | C. | -2.0 |
|  | D. | 0.0 | | E. | | 3.0 | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Question 13** | | |  | | | | D | | | | | System.out.print("Red");  System.out.println("Green\tBlue");  System.out.print("Teal"); | | |
|  | What is output by the code to the right? | | | | | | | | | | |
|  | A. | RedGreenBlueTeal | | | | | | | | | |
|  | B. | RedGreen BlueTeal | | | | | | | | | |
|  | C. | RedGreen\tBlueTeal | | | | | | | | | |
|  | D. | Red  Green Blue  Teal | | | | | | | | | |
|  | E. | RedGreen Blue Teal | | | | | | | | | |
| **Question 14** | | |  | | | | B | | | | | double value = 19.591;  System.out.printf("%5.2f", value); | | |
|  | What is output by the code to the right? | | | | | | | | | | |
|  | A. | 19 | | B. | 20.00 | | | C. | | 19.591 | |
|  | D. | 20.06 | | E. | 19.59 | | | | | | |
| **Question 15** | | |  | | | | B | | | | | public int longhorn(int y, int x){  x++;  x = y + x + y;  int z = y;  z--;  y++;  return x + y + z;  } | | |
|  | What is returned by the method call longhorn(2, 5)? | | | | | | | | | | |
|  | A. | 0 | | B. | 2 | | | C. | | 10 | |
|  | D. | 13 | | E. | 14 | | | | | | |
| **Question 16** | | |  | | | E | | | | | | | | |
|  | Which answer is logically equivalent to the following boolean expression, where w, x, y, z are int variables?  !(x == y) && !(w > z) | | | | | | | | | | | | | |
|  | A. | (x != y) && (w <= z) | | | | | | | B. | | (x != y) && (w != z) | | C. | (x > y) && (w < z) |
|  | D | (x != w) && (y != z) | | | | | | | E. | | (x < y) || (w >= z) | |  |  |
| **Question 17** | | |  | | | | B | | | | | final int LIMIT = 10;  for(int i = 0; i <= LIMIT; i++)  for(int j = 0; j < LIMIT \* 2; j++)  System.out.print('\*'); | | |
|  | How many '\*'s are output by the code to the right? | | | | | | | | | | |
|  | A. | 20 | | B. | 30 | | | C. | | 60 | |
|  | D. | 200 | | E. | 220 | | | | | | |
| **Question 18** | | |  | | | | B | | | | | String tw = "EdgarCodd";  String res = tw.substring(3);  System.out.println( res ); | | |
|  | What is output by the code to the right? | | | | | | | | | | |
|  | A. | arCodd | | B. | EC | | | C. | | odd | |
|  | D. | Edg | | E. | dd | | | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Question 19** | | |  | | | | | | | | | | D | | | | | int val = 3;  boolean p = val \* val \* val \* val > 100;  boolean q = !p;  if(!!p)  System.out.print(1);  else  System.out.print(2);  if(q || !q)  System.out.print(1);  else  System.out.print(2); | | | | |
|  | What is output by the code to the right? | | | | | | | | | | | | | | | | |
|  | A. | 11 | | | | | | | | | | | | | | | |
|  | B. | 12 | | | | | | | | | | | | | | | |
|  | C. | 21 | | | | | | | | | | | | | | | |
|  | D. | 22 | | | | | | | | | | | | | | | |
|  | E. | 121 | | | | | | | | | | | | | | | |
| **Question 20** | | |  | | | | | | | | | | B | | | | | boolean a, b, c, d;  // code to initialize a, b, c, and d  boolean e = a && b || c && d; | | | | |
|  | In the code to the right, how many of the 16 possible combinations of values for the variables a, b, c, and d will result in e being set to true? | | | | | | | | | | | | | | | | |
|  | A. | 0 | | B. | | 1 | | | | | | | | C. | 3 | | |
|  | D. | 7 | | E. | | 15 | | | | | | | | | | | |
| **Question 21** | | |  | | | | | | | | | | B | | | | | public void toy(int[] ls) {  int val = 1;  for(int i = ls.length - 1; i >= 0; i--) {  ls[i] = val \* val;  val++;  }  }  // client code  int[] data = new int[4];  toy(data);  System.out.print(Arrays.toString(data)); | | | | |
|  | What is output by the client code to the right? | | | | | | | | | | | | | | | | |
|  | A. | [16, 9, 4, 1] | | | | | B. | | | [0, 1, 2, 3] | | | | | | | |
|  | C. | [3, 2, 1, 0] | | | | | D. | | | [0, 0, 0, 0] | | | | | | | |
|  | E. | [9, 4, 1 0] | | | | | | | | | | | | | | | |
| **Question 22** | | |  | | | | | | | | | E | | | | | | | | | | |
|  | Which of the following is not a valid Java indentifier? | | | | | | | | | | | | | | | | | | | | | |
|  | A. | TV\_TIME | | | B. | | | | tvTime | | | | | | | C. | TVTIME | | D. | TV345Time | E. | TV<Time> |
| **Question 23** | | |  | | | | | | | | | | D | | | | | double seed = Math.random();  int result = ((int)(seed \* 4) - 2) \* 3;  System.out.print(result); | | | | |
|  | What are the possible values the code to the right will output? | | | | | | | | | | | | | | | | |
|  | A. | [-6, -3, 0, 3] | | | | | | | | | | | | | | | |
|  | B. | [0, 3, 6, 9, 12] | | | | | | | | | | | | | | | |
|  | C. | [-6, -3, 0, 3, 6] | | | | | | | | | | | | | | | |
|  | D. | [0, 1, 2, 3] | | | | | | | | | | | | | | | |
|  | E. | [0.0, 0.1, 0.2, 0.3, 0.4] | | | | | | | | | | | | | | | |
| **Question 24** | | |  | | | | | | | | | | B | | | | | int[] simple = {-2, 3, 1, -2};  for(int i : simple)  System.out.print(i); | | | | |
|  | What is output by the code to the right? | | | | | | | | | | | | | | | | |
|  | A. | -231-2 | | B. | | iiii | | | | | | | | C. | 0123 | | |
|  | D. | 2132 | | E. | | null null null null | | | | | | | | | | | |
| **Question 25** | | |  | | | | | | | | | | D | | | | | int[] od = new int[6];  int count = 0;  while( **<\*1>** ) {  count++;  int index = 0;  od[index]++;  int cry = 0;  do{  od[index] += cry;  cry = od[index] / 2;  od[index] %= 2;  index++;  } while(cry > 0);  }  System.out.print(count); | | | | |
|  | What replaces **<\*1>** in the code to the right to execute the body of the while loop if the element at index 5 in od is equal to 0? | | | | | | | | | | | | | | | | |
|  | A. | od[5] == 0 | | | | | | | | | | | | | | | |
|  | B. | od[5] | | | | | | | | | | | | | | | |
|  | C. | !od[5] == false | | | | | | | | | | | | | | | |
|  | D. | !od[5] | | | | | | | | | | | | | | | |
|  | E. | od[5] = 0 = true | | | | | | | | | | | | | | | |
| Assume **<\*1>** is filled in correctly. | | | | | | | | | | | | | | | | | |
| **Question 26** | | |  | | | | | | | | | | B | | | | |
|  | What is output by the code to the right? | | | | | | | | | | | | | | | | |
|  | A. | 0 | | B. | | 6 | | | | | | | | C. | 8 | | |
|  | D. | 32 | | E. | | 64 | | | | | | | | | | | |
| **Question 27** | | |  | | | | | | | | | | B | | | | | public void play(int[] vals){  vals = new int[] {2, 4, 6, 8};  for(int i = 0; i < vals.length; i++)  vals[i] += 3;  }  // client code  int[] ar = {-2, 3, -1, 1};  play(ar);  System.out.print(Arrays.toString(ar)); | | | | |
|  | What is output by the client code to the right? | | | | | | | | | | | | | | | | |
|  | A. | [-2, 3, -1, 1] | | | | | | B. | | | [2, 4, 6, 8] | | | | | | |
|  | C. | [0, 0, 0, 0] | | | | | | D. | | | [5, 7, 9, 11] | | | | | | |
|  | E. | There is no output due to a syntax error. | | | | | | | | | | | | | | | |
| **Question 28** | | |  | | | | | | | | | | B | | | | | public int tally(String evts) {  String code = "-HFT";  int total = 0;  for(int i = 0; i < evts.length(); i++) {  int pos = code.indexOf(evts.charAt(i));  switch(pos) {  case 1: total++; **<\*1>**;  case 2: total += 2; **<\*1>**;  case 3: total += 3;  }  }  return total;  } | | | | |
|  | What replaces **<\*1>** in the method tally so that subsequent case statements in the switch block are not executed for a particular character in evts? | | | | | | | | | | | | | | | | |
|  | A. | break | | B. | | goto | | | | | | | | C. | try | | |
|  | D. | throws | | E. | | discontinue | | | | | | | | | | | |
| Assume **<\*1>** is filled in correctly. | | | | | | | | | | | | | | | | | |
| **Question 29** | | |  | | | | | | | | | | B | | | | |
|  | What is returned by the method call tally("HHTFF")? | | | | | | | | | | | | | | | | |
|  | A. | 0 | | B. | | 1 | | | | | | | | C. | 4 | | |
|  | D. | 9 | | E. | | 10 | | | | | | | | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Question 30** | | |  | | | | | | | | B | | | | | int hold = 3;  int val = ++hold \* 3;  System.out.print(val + " " + hold); | | | | |
|  | What is output by the code to the right? | | | | | | | | | | | | | | |
|  | A. | 9 4 | | B. | | 9 3 | | | | | | C. | 12 4 | | |
|  | D. | 15 4 | | E. | | 12 3 | | | | | | | | | |
| **Question 31** | | |  | | | | | | | | D | | | | | public void sample(String s){  if(s.length() == 0)  System.out.print("end");  else {  System.out.print(s.charAt(0));  int len = s.length() - 1;  sample( s.substring(0, len) );  System.out.print(s.charAt(len));  }  } | | | | |
|  | What is output by the method call sample("UTA")? | | | | | | | | | | | | | | |
|  | A. | UendU | | | | | | | | | | | | | |
|  | B. | UendA | | | | | | | | | | | | | |
|  | C. | UUUendU | | | | | | | | | | | | | |
|  | D. | end | | | | | | | | | | | | | |
|  | E. | UUUendUTA | | | | | | | | | | | | | |
| **Question 32** | | |  | | | | | | | | B | | | | | public int myst(ArrayList<String> data) {  Iterator<String> it = data.iterator();  int total = 0;  while( **<\*1>** ) {  if( it.next().length() < 10 )  total += it.next().length();  }  return total;  }  // client code  ArrayList<String> sm;  sm = new ArrayList<String>();  sm.add("Moore");  System.out.print( myst(sm) ); | | | | |
|  | What replaces **<\*1>** in the method myst to execute the body of the while loop if there are any more elements in the current iteration of data? | | | | | | | | | | | | | | |
|  | A. | it.next() | | | | | B. | | it < data.size() | | | | | | |
|  | C. | i < it.size() | | | | | D. | | it.hasNext() | | | | | | |
|  | E. | None of these are correct. | | | | | | | | | | | | | |
| Assume **<\*1>** is filled in correctly. | | | | | | | | | | | | | | | |
| **Question 33** | | |  | | | | | | | | B | | | | |
|  | What is output by the client code to the right? | | | | | | | | | | | | | | |
|  | A. | 1 | | B. | | 5 | | | | | | C. | 10 | | |
|  | D. | There is no output due to a syntax error. | | | | | | | | | | | | | |
|  | E. | There is no output due to a runtime error. | | | | | | | | | | | | | |
| **Question 34** | | |  | | | | | | | E | | | | | | | | | | |
|  | Assume method sample(int[] data) is O(N2) where N = data.length. When method sample is passed an array with length = 1,000 it takes 1 second for method sample to complete. If method sample is then passed an array with length = 6,000 what is the expected time it will take method sample to complete? | | | | | | | | | | | | | | | | | | | |
|  | A. | 6 seconds | | | B. | | | 36 seconds | | | | | | C. | 60 seconds | | D. | 64 seconds | E. | 80 seconds |
| **Question 35** | | |  | | | | | | | | B | | | | | double a = 2.5;  for(int i = 0; i < 5; i++)  a++;  System.out.print(a); | | | | |
|  | What is output by the code to the right? | | | | | | | | | | | | | | |
|  | A. | 2.5 | | B. | | 5.0 | | | | | | C. | 7.5 | | |
|  | D. | There is no output due to a syntax error. | | | | | | | | | | | | | |
|  | E. | There is no output due to a runtime error. | | | | | | | | | | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Question 36** | | |  | | | | | | D | | | | byte b1 = 3;  byte b2 = 10;  byte b3 = b1 + b2; |
|  | The code to the right contains a syntax error. Which of the following best explains the reason for the syntax error? | | | | | | | | | | | |
|  | A. | b3 is not a valid identifier. | | | | | | | | | | |
|  | B. | The + operator is not defined for the type byte. | | | | | | | | | | |
|  | C. | The int literals 3 and 10 may not be assigned to variables of type byte. | | | | | | | | | | |
|  | D. | byte is not a valid Java type. | | | | | | | | | | |
|  | E. | The expression b1 + b2 evaluates to the type int and cannot be assigned to a byte variable. | | | | | | | | | | |
| **Question 37** | | |  | | | | | | B | | | | R  O  E  P  T  R |
|  | What is the result of a post order traversal of the binary tree shown to the right? | | | | | | | | | | | |
|  | A. | PORRET | | B. | | ROPTER | | C. | | | PORTER | |
|  | D. | TERROP | | E. | | ROPERT | | | | | | |
| **Question 38** | | |  | | | | | | B | | | | public class Structure<**<\*1>**> {  private ArrayList<**<\*1>**> con;  public Structure(){  con = new ArrayList<**<\*1>**>();  }  public void add(**<\*1>** val){  con.add(0, val);  }  public **<\*1>** access(){  return con.get(0);  }  public **<\*1>** remove(){  return con.remove(0);  }  public boolean isEmpty(){  return con.size() == 0;  }  } |
|  | Which of the following can replace **<\*1>** in the code to the right as the type variable for the Structure class to indicate that the Structure class is generic? | | | | | | | | | | | |
| I. II. III. | AnyType E myThings | | | | | | | | | | |
|  | A. | I only | | | B. | | II only | | | C. | | III only |
|  | D. | I and II only | | | E. | | I, II, and III | | | | | |
| Assume **<\*1>** is filled in correctly. | | | | | | | | | | | | |
| **Question 39** | | |  | | | | | | B | | | |
|  | If a Structure object contains N elements what is the Big O of the remove method in the Structure class? Pick the most restrictive correct answer. | | | | | | | | | | | |
|  | A. | O(N) | | | B. | | O(NlogN) | | | C. | | O(N3/2) |
|  | D. | O(N2) | | | E. | | O(1) | | | | | |
| **Question 40** | | |  | | | | | | B | | | |
|  | What type of data structure does the Structure class implement? | | | | | | | | | | | |
|  | A. | a list | | | B. | | a queue | | | C. | | a hashtable |
|  | D. | a heap | | | E. | | a stack | | | | | |

# 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.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.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()   **class java.util.ArrayList<E> implements List<E>**  Methods in addition to the List methods:   * 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.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.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) |

# Computer Science Answer Key

# UTCS Invitational - 2010

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | B | 11. | C | 21. | A | 31. | E |
|  | D | 12. | A | 22. | E | 32. | D |
|  | E | 13. | E | 23. | A | 33. | E |
|  | E | 14. | E | 24. | A | 34. | B |
|  | A | 15. | E | 25. | A | 35. | C |
|  | C | 16. | A | 26. | D | 36. | E |
|  | A | 17. | E | 27. | A | 37. | C |
|  | D | 18. | A | 28. | A | 38. | E |
|  | A | 19. | C | 29. | D | 39. | A |
|  | B | 20. | D | 30. | C | 40. | E |

# Notes:

The clause "Choose the most restrictive correct answer." is necessary because per the formal definition of Big O, an algorithm that is O(N2) is also O(N3) , O(N4) , and so forth.