# UIL COMPUTER SCIENCE
# S-17 Practice Packet

Written by

**Kirby Rankin**

Edited by Nancy Barnard

Author Kirby Rankin brings over 25 years of teaching experience and has coached Computer Science for most of those years. His successes include three individual champions and six 2A team champions, and these were in a row from 2008 through 2013. He had many, many more competitors qualify for region and state over his years.

*We are a small company that listens! If you have any questions or if there is an area that you would like fully explored, let us hear from you. We hope you enjoy this product and stay in contact with us throughout your academic journey.*
~ President Hexco Inc., Linda Tarrant

**HEXCO ACADEMIC**
www.hexco.com
P.O. Box 199 • Hunt, Texas 78024
Phone: 800.391.2891 • Fax: 830.367.3824
Email: hexco@hexco.com

# UIL COMPUTER SCIENCE PRACTICE PACKET

## Spring 2017

## CONTENTS

For official UIL Constitution and Contest Rules for Science, please review the Section 952 document at: http://www.uiltexas.org/academics/computerscience

# Standard Classes and Interfaces – Supplemental Reference

**`class java.long.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 vale > 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 in parseInt (String s)`
- `static int parseInt (String s, int radix)`

**`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, int 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> 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 ()`

16. Which of the following must replace **<code>** in class SomeClass shown above to ensure that the instance variables a, b, and c cannot be accessed from outside of the class?

    A. public
    B. private
    C. package
    D. this
    E. No code is necessary to restrict the access to instance variables.

17. Assume that **<code>** has been correctly replaced in the class `SomeClass`. What is the output of this client code segment?

```
int a=4,b=5,c=2,m=8,n=6;
SomeClass sc=new SomeClass(a,b,c);
sc.doSomething(m, n);
```

    A. There is no output due to an error.
    B. There is no output because `doSomething` is a void method.
    C. 4 5 2 8 6
    D. 8 6
    E. 4 5 2 20 8 14

18. Assume that **<code>** has been correctly replaced in the class `SomeClass`. What is the output of this client code segment?

```
int a=4,b=5,c=2,m=8,n=6;
SomeClass sc=new SomeClass(a,b,c);
out.print(sc.doSomethingElse(m, n)+" "+m+" "+n);
```

    A. 31 20 22
    B. 46 8 6
    C. 46 20 22
    D. 31 8 6
    E. 48 8 6

19. What is output by the code listed to the right?

```
String[] e="coding is life".split();
out.print(e.length);
```

    A. 0
    B. 1
    C. 2
    D. 3
    E. Error, will not compile

20. What is output by the code listed here?

```
Stack<Integer> s=new Stack<Integer>();
s.add(10);
s.add(7);
s.add(-5);
s.add(3);
s.pop();
out.print(s.peek()+" "+s.pop()+" "+s.get(0));
```

    A. 3 -5 10
    B. -5 7 10
    C. -5 -5 10
    D. 7 -5 3
    E. 10 7 -5

36. Which of the following is equivalent to $\overline{A+B}$?

A. $A*B+\bar{A}*\bar{B}$

B. $\bar{A}+\bar{B}$

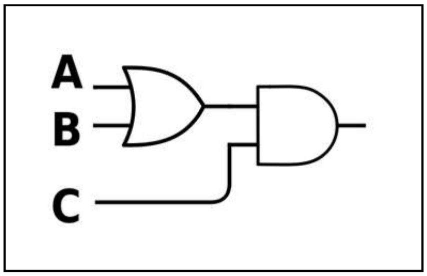
C. $\bar{A}*\bar{B}$

D. $A*B$

E. $\bar{A}*B+A*\bar{B}$

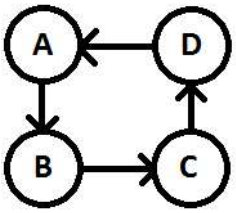
37. Which of the Boolean expressions listed is equivalent to the digital electronics diagram shown here?

A. $(A+B)+C$
B. $A*B+C$
C. $A*B+A*C$
D. $(A+B)*(A+C)$
E. None of the above.



38. What type of graph is shown in this illustration?

A. undirected and weighted
B. weighted and directed
C. undirected and unweighted
D. directed and unweighted
E. None of the above



**Questions 39 and 40 are free response type questions. Write your answers in the appropriate blanks on the answer sheet.**

39. What is the value of this postfix expression?

9 3 2 * 6 / + 9 –

40. What is the 8-bit 2’s complement representation of -25?

11. What is printed by **line #1** in the class shown below if `numbers.dat` contains 9 integers all on the same line? Assume that all required import statements are present and correct.

```
public class InputConcepts {
     public static void main(String[] args) throws
IOException{
          Scanner f=new Scanner(new File("numbers.dat"));
          int c=0;
          while(f.hasNext()){
               System.out.println(f.nextLine());
               c++;
          }
          System.out.print(c);//line #1
     }
}
```

A. 0
B. 1
C. 2
D. 9
E. 10

12. What does the code segment shown on the right print?

```
int[] nums={6,5,2,9,5,1,7,1,3,9};
int v=0,i=1;
while(i<nums.length){
     v+=nums[i];
     i+=2;}
out.print(v+" "+i);
```

A. 23 10
B. 25 10
C. 23 11
D. 25 12
E. 25 11

13. What is the correct order of operation for the operators listed to the right?

```
I. ++
II. %
III. !=
IV. +=
```

A. I II III IV
B. I II IV III
C. III II I IV
D. IV III I II
E. same level, left to right

14. Which of the following is the largest positive value that can be stored in a variable of type short without error?

A. 127
B. 32767
C. 65536
D. 2147483647
E. 9223372036854775807

15. What is printed by the code segment shown on the right?

```
ArrayList<Integer> values=new
ArrayList<Integer>();
out.print(values.size()+" ");
values.add(5);
values.add(8);
values.add(2);
out.print(values.indexOf(2)+" ");
values.set(1, 4);
out.print(values.size()+" ");
```

A. 0 3 3
B. 0 2 4
C. 0 2 3
D. null 2 3
E. Error. Size cannot be determined before values have been added to an ArrayList.

16. What is the output of the code listed to the right?

```
Set<String> s=new
TreeSet<String>();
s.add("Texans");
s.add("Cowboys");
s.add("Broncos");
s.add("Bears");
s.add("Texans");
s.remove("Bears");
out.print(s);
```

A. [Bears, Broncos, Cowboys, Texans, Texans]
B. [Texans, Cowboys, Broncos, Texans]
C. [Texans, Cowboys, Broncos]
D. [Broncos, Cowboys, Texans]
E. [Broncos, Cowboys, Texans, Texans]

17. Which of the following reserved words is used to create a user-defined constant?

A. protected
B. static
C. final
D. finally
E. private

18. What is output by the code listed to the right?

```
String s1="Texas";
String s2="Oklahoma";
out.print(s2.compareTo(s1));
```

A. -3
B. 3
C. 0
D. 5
E. -5

19. What is the output of the segment of code shown on the right?

```
int x=8,y=4,z=0;
if(x+y>15&&y/z<0)
     out.print("Yes");
else if(x>z||y>z)
     out.print("No");
else
     out.print("Maybe");
```

A. Yes
B. No
C. Maybe
D. Throws an ArithmeticException. Division by zero.
E. No output. Will not compile.