|  |  |  |
| --- | --- | --- |
| 1. Write the following for-loops as while loops | | |
|  |  | |
| int y = 5;  while(y < 10){    System.out.println(y);    y+=2;  } | int z = 10;  while(z > 0){         System.out.println(z);         z--;  } | |
|  | | /4 |

|  |  |  |
| --- | --- | --- |
| 2. What is the output for each code segment below, | | |
| (a)  int m = 0;  int j = 0;  do{  j \*= -1;  if(j >= 0){  m += 2;  }  j+=2;  }while(m < 4);  System.out.println(j); | (b)  int i = 5, j = 0;  do{  for(j = 0; j < i; j++){  System.out.print(“\*”);  }  System.out.println();  i--;  }while(i > 0); | |
| 2 | \*\*\*\*\*  \*\*\*\*  \*\*\*  \*\*  \* | |
|  | | /2 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 3. The Decrypt class below accepts a number from the user and then converts the number to its character representation. Each pair of numbers in the provided number represent the ascii equivalent of a character and therefore can be used to identify the corresponding symbol.  In the example below, each pair of numbers in num map to a different symbol as shown,   |  |  |  | | --- | --- | --- | | **num** | **pairs** | **ascii equivalents of pairs** | | 8773846772 | |  |  |  |  |  | | --- | --- | --- | --- | --- | | 87 | 73 | 84 | 67 | 72 | | |  |  |  |  |  | | --- | --- | --- | --- | --- | | W | I | T | C | H | |   Write the Decrypt class below. The final string of characters should be stored in String called result, | |
| public class Decrypt{  public static void main(String args[]){  int num = Integer.parseInt(args[0]);          String result = "";          while(num > 0){              result = (char)(num%100) + result;              num/=100;          }          System.out.println(result);  }  } | |
|  | /5 |

|  |
| --- |
| 4. A mathematical sequence is an ordered list of numbers. This question involves a sequence called a *hailstone sequence*. If n is the value of a term in the sequence, then the following rules are used to find the next term, if one exists.   * If n is 1, the sequence terminates * If n is even, then the next term is n/2 * If n is odd, then the next term is 3n + 1   For this question, assume that when the rules are applied, the sequence will eventually terminate with the term n = 1  The following are examples of hailstone sequences,  Example 1: 5, 16, 8, 4, 2, 1   * The first term is 5, so the second term is 5\*3 + 1 = 16 * The second term is 16, so the third term is 16/2 = 8 * The third term is 8, so the fourth term is 8/2 = 4 * The fourth term is 4, so the fifth term is 4/2 = 2 * The fifth term is 2, so the sixth term is 2/2 = 1 * The sixth term is 1, so the sequence terminates   Example 2: 8, 4, 2, 1   * The first term is 8, so the second term is 8/4 = 4 * The second term is 4, so the third term is 4/2 = 2 * The third term is 2, so the fourth term is 2/2 = 1 * The fourth term is 1, so the sequence terminates.   The length of a hailstone sequence is the number of terms it contains. For example, the hailstone sequence in example 1 (5, 16, 8, 4, 2, 1) has a length of 6 and the hailstone sequence in example 2 (8, 4, 2, 1) has a length of 4.  In the space below are your algorithm which calculates the length of a hailstone sequence that starts with n. |
| int n = 8;          int count = 1;          while(n > 1){              if(n % 2 == 0){                  n = n/2;              }else{                  n = 3\*n + 1;              }                count++;          }          System.out.println(count); |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 5. The CountFlips class below simulates how many flips it takes to achieve a specified streak of heads. Below are some examples,   |  |  | | --- | --- | | **Streak** | **Number of flips required to get 10 heads in a row** | | 10 | 395 | | 12 | 2648 | | 15 | 93833 |   Complete the CountFlips class below. |
| int flips = 0;          int streak = 10;          int heads = 0;          while(heads < streak){              if(Math.random() < .5){                  heads++;              }else{                  heads = 0;              }              flips++;          }          System.out.println(flips); |