**Q #1)** **How can you reverse a string?**

**Answer: String is reversed with the following algorithm:**

1. Initiate
2. The string which is to be reversed is declared.
3. Get the length of the string.
4. Start a loop and then swap the position of the array elements.
5. Keep the exchanged positions.
6. Print the reversed string.

**Q #2) What is a palindrome string?**

**Answer:**After the string is reversed as discussed in Q #1,**we need to put the following condition:**

**Code snippet:**

|  |
| --- |
| if(actualtxt.equals(reversetxt)){     return “Palindrome”;  else       return “Not Palindrome”;  } |

Thus palindrome string is the one which on reversing remains the same, **for example,** – ‘madam’ is a palindrome string.

**Q #3) How to get the matching characters in a string?**

**Answer: To get the matching characters in a string, the below steps are followed:**

1. Hash Map data structure is taken which works with the key-value pair.
2. Loop the strings, character by character, and verify if that character of the string exists in the hash map or not.
3. If the result is true, the counter for the character in the hash map is increased or else then put a count as 1.
4. Once the loop ends, then the Hash map is traversed and print the characters with more than 1 count.

**Code snippet:**

|  |
| --- |
| HashMap<Character, Integer> mp = new HashMap<> ();     for (int j = 0; j<text.length (); j++) {         char ch = text.charAt(j);            if(mp.containsKey(ch)){                  int cnt = mp.get(ch);               mp.put(ch, ++cnt);           }else{              mp.put(ch, 1);            }  }  Set<Character> charct = map.keySet();    for (Character ch: charct){       int c= mp.get(ch);       if(c>1){          System.out.println(ch+ " - " + c);       }  } |

**Q #4)** **How to get the non-matching characters in a string?**

**Answer:** **To get the non-matching characters in a string, the below steps are followed:**

1. Hash Map data structure is taken which works with the key-value pair.
2. Loop the string, character by character, and verify if that character of the string exists in the hash map or not.
3. If the result is true, the counter for the character in the hash map is increased or else then put a count as 1.
4. Once the loop ends, then the Hash map is traversed and print the characters with a count equal to 1.

**Code snippet:**

|  |
| --- |
| HashMap<Character, Integer> mp = new HashMap<> ();     for (int j = 0; j<text.length (); j++) {         char ch = text.charAt(j);            if(mp.containsKey(ch)){                  int cnt = mp.get(ch);               mp.put(ch, ++cnt);           }else{              mp.put(ch, 1);            }  }  Set<Character> charct = map.keySet();  for (Character ch: charct){       int c= mp.get(ch);       if(c==1){          System.out.println(ch+ " - " + c);       }  } |

**Q #5) How to calculate the number of vowels and consonants in a string?**

**Answer: To calculate the number of vowels and consonants in a string, the below steps are followed:**

1. Get the string on which count has to be performed.
2. Run a loop from 0 to the length of the string.
3. Take a single character at a time and verify if they are a part of the group of vowels.
4. If the result is true, increase the count of vowels or else increment the count of consonants.

**Code snippet:**

|  |
| --- |
| for (int k = 0; k < text.length(); k++) {      char c = text.charAt(k);      if (c == 'a' || c == 'e' || c == 'i' ||         c == 'o' || c == 'u')                owls += vowls  else              consonts += consonts  }  System.out.println("Vowel count is " + vowls);  System.out.println("Consonant count is: " + consonts); |

**Q #6) How do you prove that the two strings are anagrams?**

**Answer:**Two strings are called anagrams if they accommodate a similar group of characters in a varied sequence.

**To check if two strings are anagrams, the below steps are followed:**

1. Initialize two strings in two variables.
2. Check if the length of the two strings is similar, if not then the strings are not an anagram.
3. If the result is true, take the two strings and store them in a character array.
4. Sort the two character arrays, then check if the two sorted arrays are alike.
5. If the result is true, the two strings are anagram else, not anagram.

**Code snippet:**

|  |
| --- |
| if (str1.length() != str2.length()) {         System.out.println(str1 + " and " +str2 + " not anagrams string");  }else{          char[] anagram1 = str1.toCharArray();           char[] anagram2 = str2.toCharArray();           Arrays.sort(anagram1);           Arrays.sort(anagram2);            anagrmstat = Arrays.equals(anagram1, anagram2);  }  if (anagrmstat == true) {       System.out.println(str1 + " and " +str2 + " anagrams string");  }else{      System.out.println(str1 + " and " +str2 + " not anagrams string");          }  } |

**Q #7) Find the count for the occurrence of a particular character in a string.**

**Answer: To count the occurrence of a particular character in a string, the below steps are followed:**

1. Start with a string and a particular character whose occurrence shall be counted.
2. Start a loop from 0 to the length of the string.
3. Compare if a particular character of the string equals to the character that is being searched.
4. If the result is true, then increment the value of the counter.

**Code snippet:**

|  |
| --- |
| for (int l=0; l<strng.length(); l++)  {      if ( strng.charAt(l)== searchedcharacter)            rslt ++;    }  System.out.println(rslt); |

**Q #8) How to verify if two strings are a rotation mutually?**

**Answer: To verify if two strings are a rotation mutually, the below steps are followed:**

1. Initialize the two strings in two variables.
2. Check if the length of two strings is similar, if not return false.
3. Join the string to itself.
4. Verify if the string which is rotated is present in the joined string.
5. If the result is true, the second string is a rotation of the first string.

**Code snippet:**

|  |
| --- |
| String concat  = org\_string + org\_string;   if (concat.indexOf (rotat) !  = -1) {       return true;  } |

**Q #9) How to calculate the number of numerical digits in a string?**

**Answer: To calculate the number of digits in a string, the below steps are followed:**

1. Get the string on which count has to be performed
2. Use the replaceAll function, which replaces all the numerical digits with “”.
3. Get the length of the string without digits.

**Code snippet:**

|  |
| --- |
| package introduction;    public class GG {           public static void main(String[] args) {                 // TODO Auto-generated method stub                 String str = "TESTu45";                  str=str.replaceAll("\\d", "");                  int l = str.length();                  System.out.println("The length of the string without digit is:" + l);            }    } |

In this solution, a regular expression is utilized.

**Q #10) How to compute the first character of a string that is not repeated?**

**Answer: To compute the first character of a string which is not repeated, the below steps are followed:**

1. A Set data structure for the repeated characters and a list for the non-repeated character is taken.
2. After segregating the repeated and non-repeated, at the end of the iteration, the first element of the list is printed in the console.

**Code snippet:**

|  |
| --- |
| Set<Character> repeated = new HashSet<>();  List<Character> nonRepeated = new ArrayList<>();  for (int m = 0; m < wrd.length(); m++) {              char l = wrd.charAt(m);              if (repeated.contains(l)) {                  continue;              }              if (nonRepeated.contains(l)) {                  nonRepeated.remove((Character) l);                  repeated.add(l);              } else {                  nonRepeated.add(l);              }          }          return nonRepeated.get(0);      } |

**Q #11) How to search a missing number in an array that contains integers from 1 to 100?**

**Answer: To search a missing number in an array which contains integers from 1 to 100, the below steps are followed:**

1. Take an integer array with the numbers from 1 to 100.
2. Compute the sum of the numbers, the summation shall be= l\*(l+1)/2, where l is the number of integers.
3. Perform subtraction of the missing element from the total addition of numbers.

**Q #12) How to get the matching elements in an integer array?**

**Answer: To get the matching elements in an integer array, the below steps are followed:**

1. Build two loops.
2. In the first loop, collect elements one at a time and add up the number of instances of the selected element.

**Code snippet:**

|  |
| --- |
| for (m = 0; m < size; m++)          {              for (n = m + 1; n < size; n++)              {                  if (arry[m] == arry[n])                      System.out.print(arr[m]);              }          } |

**Q #13) How to delete the repeated elements in an integer array?**

**Answer: To delete the repeated elements in an integer array, the below steps are followed:**

1. Build a hashmap that will pick all the elements that are present before.
2. Loop through the array and verify if the element already exists in the hash map
3. If the result is true, array traversal is continued, otherwise, the element is printed out in the console.

**Code snippet:**

|  |
| --- |
| HashMap<Integer,Boolean> m =  new HashMap<>();      for (int j = 0; j < a.length); j++)           {              if (m.get(a[j]) == null)                  System.out.print(a[j] + " ");                   mp.put(a[j], true);          }      } |

**Q #14) Determine the largest and the smallest element of an array which is not sorted.**

**Answer: To determine the largest and the smallest element of an array the below steps need to be followed:**

1. Traverse the array, and monitor the maximum element found so far, till we are at the border of the array, the largest element is achieved.
2. Traverse the array, and monitor the minimum element found so far, till we are at the border of the array, the smallest element is achieved.

**Q #15) Explain the bubble sort algorithm.**

**Answer: The bubble sort algorithm includes the following steps:**

1. Begin from the first element, then perform a comparison with the following element in the array
2. If the present element is larger than the following element of the array, then swap their positions.
3. If the present element is lesser than the following element of the array, shift to the next element, and again repeat step 1.

**Code snippet:**

|  |
| --- |
| for(k = 0; k < arry.length; k++)      {          for(l = 0; l < arry.length-l-1; l++)          {              if( arry[l] > arr[l+1])              {                  t = arry[l];                  arry[l] = arry[l+1];                  arry[l+1] = t;              }          } |

**Q #16) Implement the insertion sort algorithm.**

**Answer:**Implementation of insertion sort.

**Code snippet:**

|  |
| --- |
| for (m = 1; m < arry.length; m++)      {          n = m;          while (n> 0 && arry[n - 1] > arry[n])          {              k = arry[n];              arry[n] = arry[n - 1];              arry[n - 1] = k;              n--;          }      } |

**Q #17) Determine the second largest element of an array.**

**Answer: The second largest element of an array can be computed by the following steps:**

1. State the largest element as the first element of the array and the second largest element as the second element of the array.
2. Iterate through the loop for traversing the array.
3. IF arry[i] is greater than the largest element THEN  
   Second element ? largest element  
   Largest element ?arry[i]  
   IF second element is less than arry[i] THEN  
   Second element?arry[i]

**Code snippet:**

|  |
| --- |
| if(arry[0] > arry[1]) {        l = arry[0];        s  = arry[1];     } else {        l = arry[1];        s = arry[0];     }       for(i = 2; i < arry.length; i++) {        if( l < a[i] ) {           s = l;           l = arry[i];        } else if( s < arry[i] ) {           s =  arry[i];        }     } |

**Q #18) Explain the reversal of an array.**

**Answer: Array reversal is performed in the following ways:**

1. Take an array with elements.
2. Now exchange the position of the first element with the final element, and similarly the second element with the penultimate element.
3. This will continue until the entire array is reversed.

**Code snippet:**

|  |
| --- |
| for (t = 0; t < arr.length / 2; t++) {              tmp = arr[t];              arr[t] = arr[arr.length - t - 1];              arr[arr.length - t- 1] = tmp;          } |

**Q #19) How to remove special characters in a string that is in lowercase?**

**Answer:**Special characters in a string can be removed by using the replaceAll function in Java.

**Code snippet:**

|  |
| --- |
| string str = “Testing@”  str.replaceAll(“[^a-z0-9]”,””) |

In this solution, a regular expression is utilized.

**Q #20) How to perform swapping two strings by not using a third variable?**

**Answer: Two strings are swapped without the help of the third variable by the following steps:**

**(i)** Take two strings i, j, and append them then store in the first string.

**(ii)** Using the substring method extract the string:

j = substring(0,i.length()-j.length())

**(iii)** Store the string j in string i

i= subsrtirng(j.length)

**Code snippet:**

|  |
| --- |
| string i = “abc”, j =”def”;      i = i+j;      j = i. substring(0,i.length()-j.length());      i = i.substring(j.length())     System.out.println(i +””+j); |

**Q #21) How to traverse to the middle of a linked list?**

**Answer: To traverse to the middle of a linked list the below steps are followed:**

1. Declare two pointers first and second which are initialized to the linked list head.
2. Increment the first linked list by two nodes and second by one node in each loop.
3. While the first node reaches the end of the list, the second node will point to the middle.

**Code snippet:**

|  |
| --- |
| first = second = head;   while(first !=null) {      first = first.next;       if(first != null && first.next != null) {            second = second.next;            first = first.next;       }  }   return second;    } |

**Q #22) Implement the process of reversing a linked list.**

**Answer: A linked list can be reversed by the below steps:**

1. Declare three nodes preceding, present, and following.
2. While in the present node, the preceding will be null.
3. Let the present.next be preceding to reverse the list.
4. In each looping, present and preceding are incremented by 1.

**Code snippet:**

|  |
| --- |
| Node preceding=null;  Node following  while(present!=null)   {       following=present.next;       present.next=preceding;       preceding=present;       present=following;    }  return preceding;  } |

**Q #23) What is the process of deleting matched elements from a linked list which is not sorted.**

**Answer: To delete matched elements from a linked list which is not sorted, the below steps are followed:**

1. Travel from the head to the tail of the linked list.
2. For every value in the linked list, verify if it’s already present in the hash table.
3. If the result is true, the element is not added to the hash table.

**Code snippet:**

|  |
| --- |
| HashSet<Integer> h = new HashSet<>();            node present = head;          node preceding = null;          while (present != null)          {              int presentval = present.value;                if (h.contains(presentval)) {                  preceding.next = present.next;              } else {                  h.add(presentval);                  preceding = present;              }              present = present.next;          }        } |

**Q #24) How to get the length of a linked list?**

**Answer:** **To get the length of a linked list, the below steps are followed:**

1. Start a counter with a value of 0 and present node as head.
2. Till the present node is not null , perform these :
   * present = present -> next
   * counter = counter + 1
3. Counter value is returned.

**Code snippet:**

|  |
| --- |
| {          Node present = head;          int c = 0;          while (present != null)          {              c = c + 1;              present = present.next;          }          return c;      } |

**Q #25) How to search a specific value in a linked list?**

**Answer: To search a specific value in a linked list, the below steps are followed:**

1. Declare present node as head.
2. Till the present node is not null, perform these :
   * present -> value is equal to the value being looked for return true.
   * present = present -> next.
3. If not found, false is returned.

**Code snippet:**

|  |
| --- |
| Node present = head;          while (present != null)          {              if (present.value == value)                  return true;              present = present.next;          }          return false;      } |

**Q #26) How to verify if a number is prime or not?**

**Answer:** **To verify if a number is prime or not, the below steps are followed:**

1. Start a loop from value 2(k) up to (number / 2)
2. If the number is perfectly divisible by k, then the number is non – prime.
3. If the number is not perfectly divisible except for 1 and by itself, then the number is prime.

**Code snippet:**

|  |
| --- |
| for(k = 2; k <= number / 2; k++)    {        if(number % k == 0)        {            stat = false;            break;        }    }    if (stat)       System.out.println("Prime";      else        System.out.println("Not prime"); |

**Q #27) How to get the third node of a linked list?**

**Answer: To get to the third node of the linked list the below steps are followed:**

1. Start a counter with a value 0.
2. Iterate through the linked list and perform these steps:
   * If the value of the counter is 3, then the present node is returned.
   * Counter is increased by 1.
   * Modify the present such that it implies to the next of the present.

**Code snippet:**

|  |
| --- |
| Node present = head;          int c = 0;          while (c != null)          {              if (c == 3)                  return present.val;              c = c+1;              present = present.next;          } |

**Q #28) Compute the first five Fibonacci numbers.**

**Answer:**0 and 1 are the first two Fibonacci numbers and all the numbers after 0 and 1 are the addition of the two previous numbers.

**Code snippet:**

|  |
| --- |
| int num1=0, num2=1,t;  for ( int k = 0; k<5,k++)  {     System.out.println(num1);     t = num1 + num2;     num1 = num2; num2 = t;  } |

**Q #29) How to reverse a number?**

**Answer: Reversal of a number is achieved in the following steps:**

1. Take out the rightmost digit of the number.
2. Sum up the digit with the new reversed number.
3. Perform multiplication by 10.
4. Divide the number by 10.

**Q #30) Determine the factors of a number.**

**Answer: The factors of a number is expressed by the following code snippet:**

|  |
| --- |
| int no = 75;             for(int j = 1; j <= no; j++) {              if (no % j == 0) {                  System.out.print(j);              } |

**Conclusion**