Assignment_Quiz2

Email *

Name *

Ss

Email *

rathsmita84@gmail.com

Regno *

344

Section

Fgg

Quiz Questions

if the condition to group elements larger than pivot is true swapping of elements takes place between * 2 points

```
public static enum Color { RED, WHITE, BLUE }

public static void dutchFlagPartition(int pivotIndex, List<Color> A) {
  Color pivot = A.get(pivotIndex);
  // First pass: group elements smaller than pivot.
  int smaller = 0;
  for (int i = 0; i < A.size(); ++i) {
    if (A.get(i).ordinal() < pivot.ordinal()) {
      Collections.swap(A, smaller++, i);
    }
  }
  // Second pass: group elements larger than pivot.
  int larger = A.size() - 1;
  for (int i = A.size() - 1; i >= 0 && A.get(i).ordinal() >= pivot.ordinal();
       --i) {
    if (A.get(i).ordinal() > pivot.ordinal()) {
```

- ○ Collections.swap (A, i ,larger++)
- ◉ Collections.swap(A , larger--, i);
- ○ Collections.swap(A , smaller++, larger--);
- ○ Collections.swap(A , larger--, smaller++);

What is time complexity and space complexity of above code to partition the array A? * 2 points

O(n) O(1)

What is the code if (A.get(equal).ordinal() < pivot.ordinal()){ _____} where equal is same as mid and smaller is same as low and larger is same as high positions in the Array A. * 2 points

```
public static enum Color { RED, WHITE, BLUE }

public static void dutchFlagPartition(int pivotIndex, List<Color> A) {
  Color pivot = A.get(pivotIndex);

  /**
   * Keep the following invariants during partitioning:
   * bottom group: A.subList(0, smaller).
   * middle group: A.subList(smaller, equal).
   * unclassified group: A.subList(equal, larger).
   * top group: A.subList(larger, A.size()).
   */
  int smaller = 0, equal = 0, larger = A.size();
  // Keep iterating as long as there is an unclassified element.
  while (equal < larger) {
    // A.get(equal) is the incoming unclassified element.
```

- (●) Collections.swap(A , smaller++, equal++);
- ( ) ++equal ;
- ( ) Collections.swap(A , equal, --larger);
- ( ) ++smaller;

What is the code if (A.get(equal).ordinal() ==pivot.ordinal()){ _____} where equal is same as mid and smaller is same as low and larger is same as high positions in the Array A. 2 points

- ( ) Collections.swap(A , smaller++, equal++);
- (●) ++equal ;
- ( ) Collections.swap(A , equal, --larger);
- ( ) ++smaller;

What is the code if (A.get(equal).ordinal() > pivot.ordinal()){ _____} where equal is same as mid and smaller is same as low and larger is same as high positions in the Array A. 2 points

- [x] Collections.swap(A , equal, --larger);
- [ ] ++smaller;
- [ ] ++equal ;
- [ ] Collections.swap(A , smaller++, equal++);

If the result has an additional digit, e.g., 99 + 1 = 100, all digits have to be moved to the right by one. Then the code that is executed to store 100 in array A when input A =[9 9] ie, if (A . get (0) == l0) {// Need additional digit as the most significant digit (i.e A. get (9))// has a carry-out . 2 points

- [ ] A .set (0 ,10) ;A . add (0 , 1);
- [x] A .set (0 ,0) ;A . add (0 , 1);
- [ ] A .set (1 ,9) ;A . add (0 , 1);
- [ ] A .set (0 ,0) ;A . add (0 , 10);

what is the result array updated when code is run once for i=num1.size()-1 and for(j=num2.size()-1;j>=0;j--) when num1=123 and num2 986? 2 points

```
List<Integer> result
    = new ArrayList<>(Collections.nCopies(num1.size() + num2.size(), 0));
for (int i = num1.size() - 1; i >= 0; --i) {
  for (int j = num2.size() - 1; j >= 0; --j) {
    result.set(i + j + 1,
               result.get(i + j + 1) + num1.get(i) * num2.get(j));
    result.set(i + j, result.get(i + j) + result.get(i + j + 1) / 10);
    result.set(i + j + 1, result.get(i + j + 1) % 10);
  }
}
```

- [0 0 0 0 0 18]
- [0 0 0 0 1 8]
- [0 0 0 0 13 8]
- [0 0 0 1 3 8]
- [0 0 0 7 3 8]

What is the time complexity for the above code for multiplying two nos num1 and num2? 2 points

O(nm)

If A = (3, 2,0,0, 2,0,1},we iteratively update the furthest we can advance to calling the below function canReachEnd(A) what is the value of furthestReachSoFar? 2 points

```
public static boolean canReachEnd(List<Integer> maxAdvanceSteps) {
  int furthestReachSoFar = 0, lastIndex = maxAdvanceSteps.size() - 1;
  for (int i = 0; i <= furthestReachSoFar && furthestReachSoFar < lastIndex;
       ++i) {
    furthestReachSoFar
        = Math.max(furthestReachSoFar, i + maxAdvanceSteps.get(i));
  }
  return furthestReachSoFar >= lastIndex;
}
```

- (●) 3
- ( ) 4
- ( ) 5
- ( ) 6

Find the value of maxProfit if array of stock prices: (310,315, 275, 295, 260, 270, 290, 230, 255, 250) 2 points

```
public static double computeMaxProfit(List<Double> prices) {
  double minPrice = Double.MAX_VALUE, maxProfit = 0.0;
  for (Double price : prices) {
    maxProfit = Math.max(maxProfit, price - minPrice);
    minPrice = Math.min(minPrice, price);
  }
  return maxProfit;
}
```

- ( ) 20
- ( ) 25
- (●) 30
- ( ) 35

This form was created inside of SIKSHA 'O' ANUSANDHAN.

Google Forms