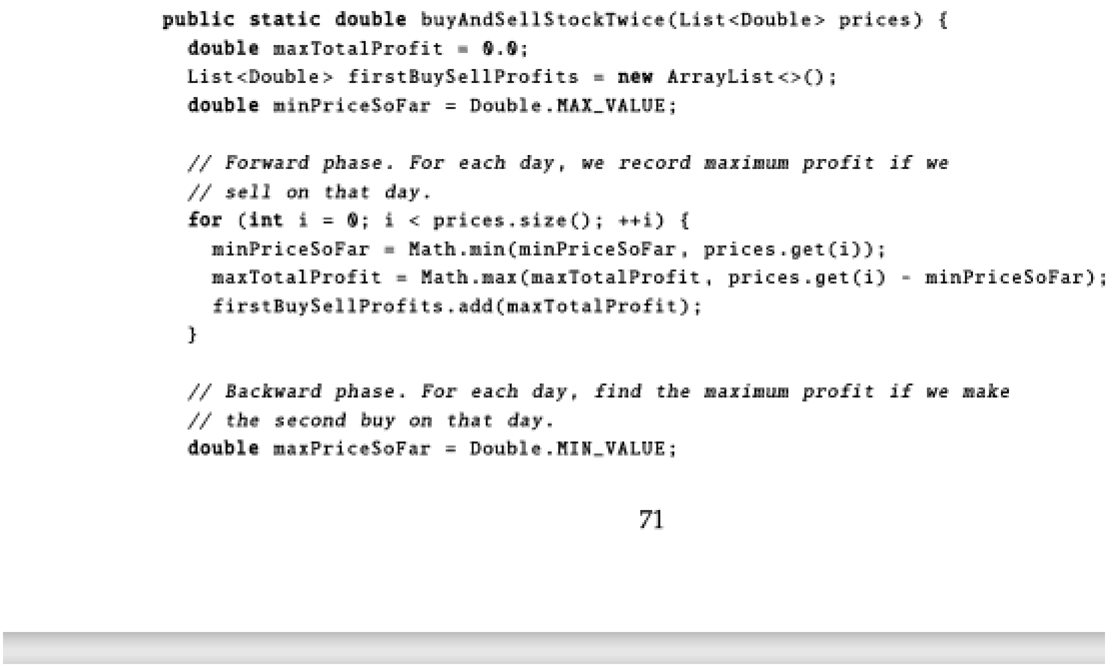
Find the maximum profit if Array A =[12,11,13,9,12,8,14,13,15] after tracing the function given in the figure. 3 points

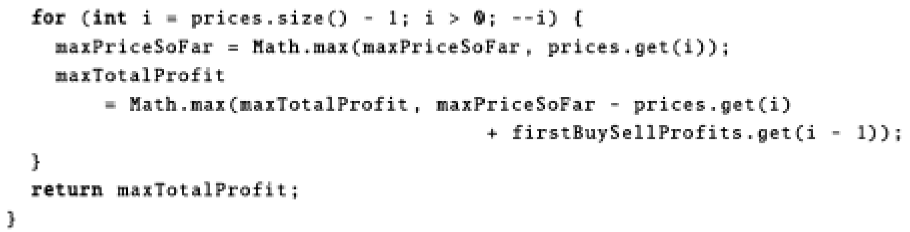


```
public static double buyAndSellStockTwice(List<Double> prices) {
  double maxTotalProfit = 0.0;
  List<Double> firstBuySellProfits = new ArrayList<>();
  double minPriceSoFar = Double.MAX_VALUE;

  // Forward phase. For each day, we record maximum profit if we
  // sell on that day.
  for (int i = 0; i < prices.size(); ++i) {
    minPriceSoFar = Math.min(minPriceSoFar, prices.get(i));
    maxTotalProfit = Math.max(maxTotalProfit, prices.get(i) - minPriceSoFar);
    firstBuySellProfits.add(maxTotalProfit);
  }

  // Backward phase. For each day, find the maximum profit if we make
  // the second buy on that day.
  double maxPriceSoFar = Double.MIN_VALUE;
```

71



```
  for (int i = prices.size() - 1; i > 0; --i) {
    maxPriceSoFar = Math.max(maxPriceSoFar, prices.get(i));
    maxTotalProfit
        = Math.max(maxTotalProfit, maxPriceSoFar - prices.get(i)
                                   + firstBuySellProfits.get(i - 1));
  }
  return maxTotalProfit;
}
```

○ 7

● 10

○ 9

○ 6

In the above question ,Print the values of maxTotalProfit at each iteration i in an array P. 2 points

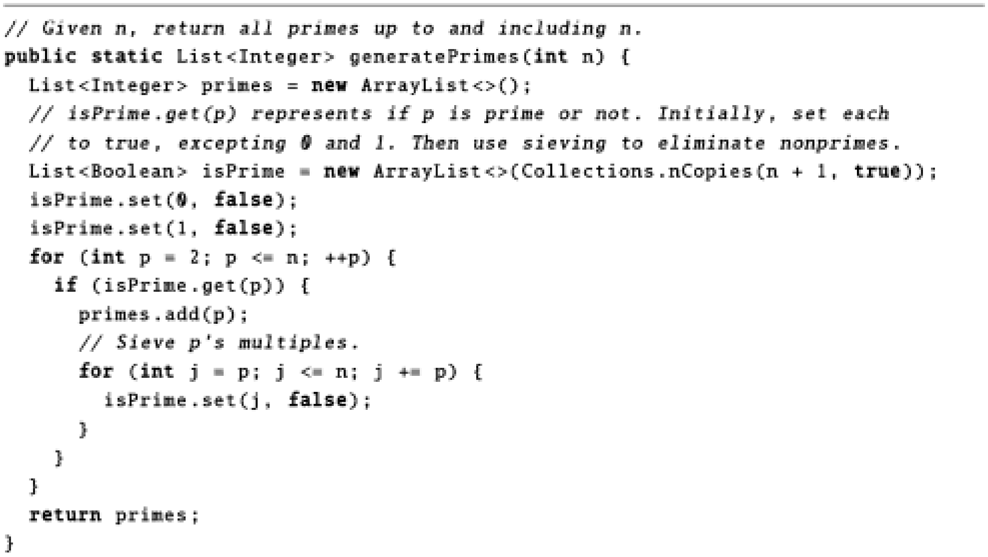
● P = [7, 7,7,9,9,10,5,8,6]

○ P = [7, 7,7,9,9,9,5,8,7]

○ P = [7, 7,7,9,7,5,5,8,7]

○ P = [6, 6,6,6,6,5,5,6,6]

Print the array isPrime for n=20. * 2 points



```
// Given n, return all primes up to and including n.
public static List<Integer> generatePrimes(int n) {
  List<Integer> primes = new ArrayList<>();
  // isPrime.get(p) represents if p is prime or not. Initially, set each
  // to true, excepting 0 and 1. Then use sieving to eliminate nonprimes.
  List<Boolean> isPrime = new ArrayList<>(Collections.nCopies(n + 1, true));
  isPrime.set(0, false);
  isPrime.set(1, false);
  for (int p = 2; p <= n; ++p) {
    if (isPrime.get(p)) {
      primes.add(p);
      // Sieve p's multiples.
      for (int j = p; j <= n; j += p) {
        isPrime.set(j, false);
      }
    }
  }
  return primes;
}
```

○ isPrime=[F F T T T T F T F F F T F T F F F T F T F]

● isPrime=[F F T T F T F T F F F T F T F F F T F T F]

○ isPrime=[F F T T F T F T F T F T F T F F F T F T F]

○ isPrime=[F F F F F T F T F F F T F T F F F T F T F]

To reduce the size of the array of isPrime array in above question what is the new value of n is

1 point

○ size = (int)Math.floor(1/2* (n - 3)) + 1;

○ n = (int)Math.floor((n - 3)) + 1;

● n = (int)Math.floor(0.5 * (n - 3)) + 1;

To print the values of prime nos what is the value of next prime no if(isPrime.get(i)==true) in the above program as now size is reduced and assume prime.add(2) initially.

2 points

(2*i+3)

To sieve out the multiples of p what is the value of j in the for loop?

2 points

● j = ((i *i)*2)+6*i + 3 + p

○ j=p+2

○ j=j+p

To apply it to an array A = {a,b,c,d),we move the element at index 0 (a) to index 3 and the element already at index 3 (6) to index 0. Continuing, we move the element at index1(b) to index 2 and the element already at index 2 (c)to index 1. Now all elements have been moved according to the permutation, and the result is(d, c, b, a).Then the permutation array P applied to it is ____________

2 points

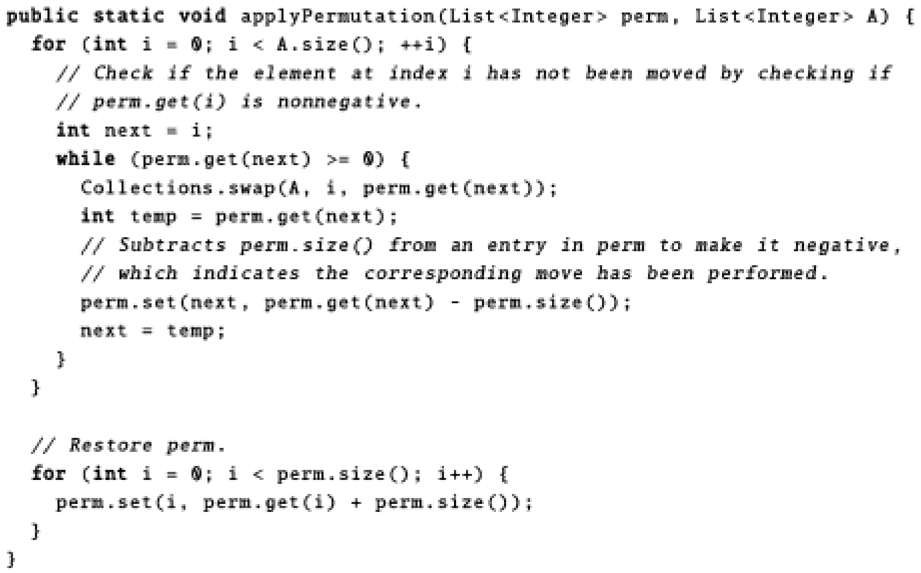
● P={3, 2,1,0}.

○ P={2,3,1,0}.

○ P={3, 2,0,1}.

Apply the above array A and P=[3,1,2,0] to this program and print the resultant P array before restoring again .

2 points



```
public static void applyPermutation(List<Integer> perm, List<Integer> A) {
  for (int i = 0; i < A.size(); ++i) {
    // Check if the element at index i has not been moved by checking if
    // perm.get(i) is nonnegative.
    int next = i;
    while (perm.get(next) >= 0) {
      Collections.swap(A, i, perm.get(next));
      int temp = perm.get(next);
      // Subtracts perm.size() from an entry in perm to make it negative,
      // which indicates the corresponding move has been performed.
      perm.set(next, perm.get(next) - perm.size());
      next = temp;
    }
  }

  // Restore perm.
  for (int i = 0; i < perm.size(); i++) {
    perm.set(i, perm.get(i) + perm.size());
  }
}
```

○ [3,-2,1,-4]

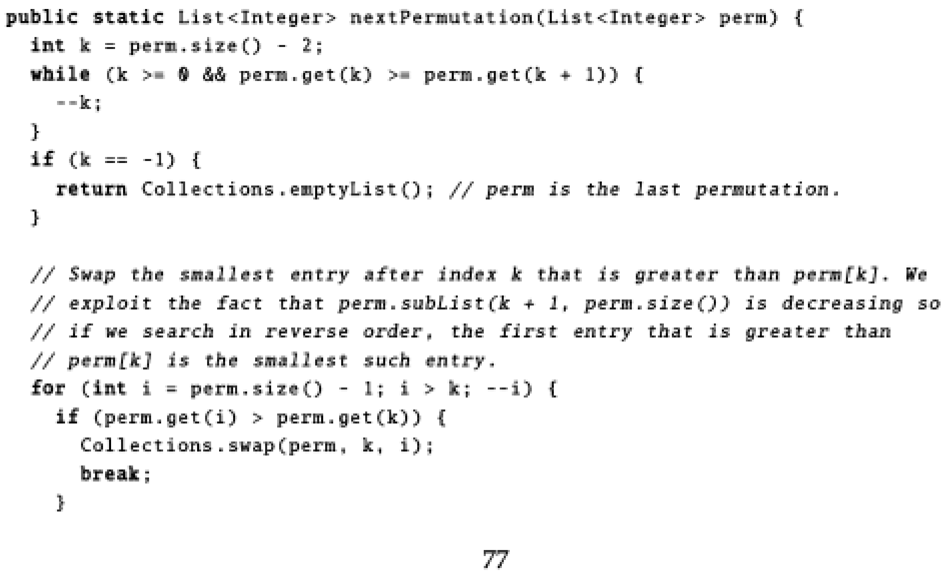
○ [-1,1,2,0]

○ [-1,1,-2,-4]

● [-1,-3,-2,-4]

What is the next permutation found after tracing the program nextPermutation(p) where p=[6,2,1,5,4,3,0]

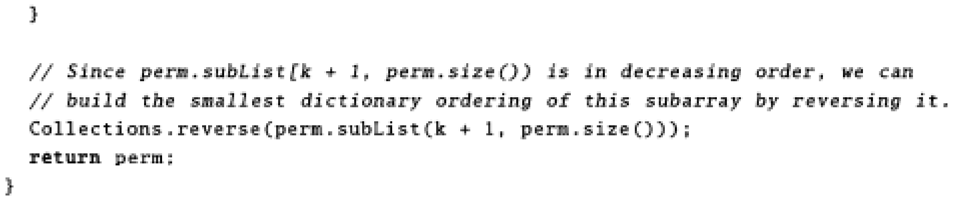
2 points



```
public static List<Integer> nextPermutation(List<Integer> perm) {
  int k = perm.size() - 2;
  while (k >= 0 && perm.get(k) >= perm.get(k + 1)) {
    --k;
  }
  if (k == -1) {
    return Collections.emptyList(); // perm is the last permutation.
  }

  // Swap the smallest entry after index k that is greater than perm[k]. We
  // exploit the fact that perm.subList(k + 1, perm.size()) is decreasing so
  // if we search in reverse order, the first entry that is greater than
  // perm[k] is the smallest such entry.
  for (int i = perm.size() - 1; i > k; --i) {
    if (perm.get(i) > perm.get(k)) {
      Collections.swap(perm, k, i);
      break;
    }
```

77



```
    }

    // Since perm.subList[k + 1, perm.size()) is in decreasing order, we can
    // build the smallest dictionary ordering of this subarray by reversing it.
    Collections.reverse(perm.subList(k + 1, perm.size()));
    return perm;
  }
```

○ p=[6,2,1,5,4,3,0]

○ p=[6,2,3,5,4,1,0]

● p=[6,2,3,0,1,4,5]

let the input be A = (3, 7,5,11} and the size be 3. In the first iteration, we use the random number generator to pick a random integer in the interval [0,3], Let the returned random number be 2. We swap A[0] with A[2]— now the array is (5,7,3,11). Now we pick a random integer in the interval [1,3].Let the returned random number be 3. We swap A[I] with A[3]—now the resulting array is (5,11,3,7). Now we pick a random integer in the interval [2,3]. Let the returned random number be 2. When we swap A[2] with itself the resulting array is unchanged. The random subset consists of the first three entries, i.e., (5,11,3}.Write equation to generate the random no as per the iteration.

2 points

i + gen . nextInt (A.size () - i));

The four intervals are [0.0, 0.5),[0.5,0.833),[0.833,0.944),[0.944, 1.0], Now, for example,if the random number generated uniformly in [0.0, 1.0] is 0.873,Then random no can be generated using double n =________________.

2 points

r . nextDouble () ;

For example, suppose n = 100 and k = 4. In the first iteration, suppose we get the random number 28. We update H to (0, 28),(28,0). This means that A[0] is 28 and A[28] is 0—for all other i, A[i] = i. In the second iteration ,suppose we get the random number 42. We update H to (0, 28),(28,0),(1, 42),(42,1). In the third iteration, suppose we get the random number 42 again and then random no next is 64.Then what is next update in H=[(keys,values)]

3 points

(0,28),(1,42),(2,1),(3,64),(28,0),(42,2),(64,3)

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

Google Forms