Siddhant Jaiswal
1941012646
CSE - F

CSW - ASSIGNMENT 2

Q1. Write a program to implement dutch national flag partition.

Code:

```
import java.util.*;
 class DutchNationalFlag {
    public static void dfp(List<Integer> arr) {
        int b=0,m=0,pivot=1,e=arr.size()-1;
        while(m<=e) {
            if(arr.get(m)<pivot) {
                Collections.swap(arr,b, m);
                ++b;
                ++m;
            }
            else if(arr.get(m)>pivot) {
                Collections.swap(arr, m, e);
                --e;
            }
            else {
                ++m;
            }
        }
    }

    public static void main(String[] args) {
        List<Integer> al=new ArrayList<Integer>();
        Collections.addAll(al, 0,1,2,0,2,1,1);
        dfp(al);
        System.out.println("List "+al);
    }

}
```

Output

```
List [0, 0, 1, 1, 1, 2, 2]
```

2. Write a program which takes as input an array of digits encoding a decimal number D and update the array to represent the number D + 1.

CODE:

```
import java.util.*;
public class DigitsEncoding {
    public static List<Integer> plusOne (List<Integer>
l) {
        int n=l.size()-1;
        l.set(n,l.get(n)+1);
        for (int i=n;i>0&&l.get(i)==10;i--) {
            l .set(i,0) ;
            l.set(i-1,l.get(i-1)+1);
        }
        if (l.get(0)==10) {
            l.set(0,0);
            l.add(0,1);
        }
        return l ;
    }
    public static void main(String[] args) {
        List<Integer> l=new ArrayList<Integer>();
        l.add(1);
        l.add(2);
        l.add(9);
        System.out.println(plusOne(l));
    }

}
```

Output :

```
[1, 3, 0]
```

3 .Write a program that takes two arrays representing integer, and return an integer representing their product.

Code :

```
import java.util.*;
public class IntegerProduct {
    public static List <Integer> multiply (List<Integer>l1,List<Integer>l2) {
        int sign=l1.get(0)<0^l2.get(0)<0?-1:1;
        l1.set(0,Math.abs(l1.get(0)));
        l2.set(0,Math.abs(l2 .get(0)));
        List<Integer>res=new ArrayList<Integer>(Collections.nCopies(l1.size()+l2.size(),0));
        for(int i=l1.size()-1;i>=0;--i){
            for(int j=l2.size()-1;j>=0;--j){
                res.set(i+j+1,res.get(i+j+1)+l1.get(i)*l2.get(j));
                res.set(i+j,res.get(i+j)+res.get(i+j+1)/10);
                res.set(i+j+1,res.get(i+j+1)%10);
            }
        }
        int non0=0;
        while (non0<res.size()&&res.get(non0)==0)
            non0++;
        res=res.subList(non0,res.size());
        if (res.isEmpty())
            return Arrays.asList(0);
        res.set(0,res.get(0)*sign);
        return res;
    }
    public static void main(String[] args) {
        List<Integer> l=new ArrayList<Integer>();
        List<Integer> l1=new ArrayList<Integer>();
        Collections.addAll(l,1,2,3);
        Collections.addAll(l1,4,5,6);
        System.out.println(multiply(l,l1));
    }

}
```

Output :

```
[5, 6, 0, 8, 8]
```

4. Write a program which takes as input a sorted array and updates it so that all duplicate have been removed and remaining elements shifted left to fill the emptied indices.

Code :

```
import java.util.*;
public class A2qno4 {
    public static int deleteDuplicates(List<Integer>l) {
        int j=0;
        for (int i=0;i<l.size()-1;i++)
            if (l.get(i)!=(l.get(i+1)))
                l.set(j++,l.get(i));
            l.set(j++,l.get(l.size()-1));
        return j;
    }
    public static void main(String[] args) {
        List<Integer> l=new ArrayList<Integer>();
        Collections.addAll(l,2,3,5,7,11,11,11,13);
        int n=deleteDuplicates(l);
        for(int i=0;i<n;i++) {
            System.out.print(l.get(i)+" ");
        }
    }
}
```

Output :

```
2 3 5 7 11 13
```

5. Write a program that takes an array denoting the daily stock price, and return the maximum profit that could be made by buying and then selling one share of that stock.

Code :

```
import java.util.*;
public class StockMax{
    public static int profit(List<Integer>l) {
        String mi="",ma="";
        int min=Integer.MAX_VALUE,max=0;
        for(int p:l) {
            min=Math.min(min, p);
            mi+=min+" ";
            max=Math.max(max, p-min);
            ma+=max+" ";
        }
        System.out.println(mi);
        System.out.println(ma);
        return max;
    }
    public static void main(String[] args) {
        List<Integer> l=new ArrayList<Integer>();
        Collections.addAll(l,310,315,275,295,260,270,290,230,255,250);
        System.out.println(profit(l));
    }

}
```

Output :

```
310 310 275 275 260 260 260 230 230 230
0 5 5 20 20 20 30 30 30 30
30
```

6 . Write a program that computes the maximum profit that can be made by buying and selling a share at most twice. The second by must be made on another date after the first sale.

Code :

```
import java.util.*;
public class StockTwice {
    public static int profit(List<Integer> p) {
        int tmaxprofit=0;
        List<Integer> fprofits=new ArrayList<Integer>();
        int minprice=Integer.MAX_VALUE;
        for(int i=0;i<p.size();++i){
          minprice=Math.min(minprice,p.get(i));
          tmaxprofit=Math.max(tmaxprofit,(p.get(i)-minprice));
          fprofits.add(tmaxprofit);
        }

        int maxprize=Integer.MIN_VALUE;
        for(int i=p.size()-1;i>0;--i)
        {
          maxprize=Math.max(maxprize,p.get(i));
          tmaxprofit=Math.max(tmaxprofit,maxprize-p.get(i)+fprofits.get(i-1));
        }
        return tmaxprofit;
    }
    public static void main(String[] args) {
        List<Integer> l=new ArrayList<Integer>();
        Collections.addAll(l,12,11,13,9,12,8,14,13,15);
        System.out.println(profit(l));
    }

}
```

Output :

```
10
```

7 .Write a program that takes an integer argument and return all the primes between 1 and the integer.

Code :

```java
import java.util.*;
public class Primes {
    public static  List<Integer> generatePrimes(int n){
        List<Integer> primes=new ArrayList<>();
        int c=0;
        for(int i=2;i<=n;i++) {
            c=0;
          for(int j=2;j*j<=i;j++) {
            if(i%j==0)
              c++;
          }
          if(c==0) {
            primes.add(i);
          }
        }
        return primes;
    }
    public static void main(String[] args) {
        System.out.println(generatePrimes(10));
    }

}
```

Output :

```
[2, 3, 5, 7]
```

8 . Given an array A of n elements and a permutation P, apply P to A.

Code :

```java
import java.util.*;
public class Permutation{
    public static void applyPermutation(List<Character> A,List<Integer> P
){
        for (int i =0;i<A.size();++i){
            int next=i;
            while (P.get(next)>=0){
                Collections.swap(A,i,P.get(next));
                int temp = P.get(next);
                P.set(next,P.get(next)-P.size());
                next = temp;
            }
        }
    }
    public static void main(String[] args) {
        List<Character> l=new ArrayList<>();
        List<Integer> l2=new ArrayList<>();
        Collections.addAll(l,'a','b','c','d');
        Collections.addAll(l2,2,0,1,3);
        applyPermutation(l,l2);
        System.out.println(l);
    }

}
```

Output :

```
[b, c, a, d]
```

9 .Write a program that takes as input a permutation, and returns the next permutation under dictionary ordering. If the permutation is the last permutation, return the empty array. For example, if the input is < 1,0,3,2 > your function should return < 1,2,0,3 >. If the input is < 3,2,1,0 >, return ().

Code:

```
import java.util.*;
public class A9{
    public static List<Integer> nextPermutation(List<
Integer> l) {
        int k=l.size()-2;
        while(k>=0&&l.get(k)>=l.get(k+1)) {
            k--;
        }
        if (k==-1) {
        return Collections.emptyList();
        }
        for (int i=l.size()-1;i>k;--i) {
            if (l.get(i)>l.get(k)) {
                Collections.swap(l, k, i);
                break ;
            }
        }
        Collections.reverse (l.subList(k+1,l.size())) ;
        return l;
    }
    public static void main(String[] args) {
        List<Integer> l=new ArrayList<Integer>();
        Collections.addAll(l,65,66,67);
        nextPermutation(l);
        System.out.println(l);
    }

}
```

Output :

```
[65, 67, 66]
```