Question 3

1)

$$T(n) \leq \begin{cases} n & for\, n \leq 2 \\ 3.T\left(\left\lceil\frac{n}{2}\right\rceil\right) + 6n & for\, n > 2 \end{cases}$$

$$T_k(n) = \left[3.T\left(\frac{n}{2^k}\right) + 6n\right]$$

$T_2(n) = 3\left[3.T\left(\frac{n}{2^2}\right) + 6\left(\frac{n}{2}\right)\right] + 6n$ => $T_2(n) = 3^2T\left(\frac{n}{2^2}\right) + (1 + \frac{3}{2})\, 6n$

$T_3(n) = 3^2\left[3.T\left(\frac{n}{2^2}\right) + 6\left(\frac{n}{2}\right)\right] + (1 + \frac{3}{2})\, 6n$ => $T_2(n) = 3^3T\left(\frac{n}{2^3}\right) + (1 + \frac{3}{2} + (\frac{3}{2})^2)\, 6n$

$$T_k(n) = 3^kT(\frac{n}{2^k}) + \sum_{i=0}^{k-1}[\frac{3}{2}]\, 6n$$

$$3^kT(\frac{n}{2^k}) = O(nlog\, n) \quad \sum_{i=0}^{k-1}[\frac{3}{2}]\, 6n = O(n)$$

$$O(nlog\, n)\ is\ the\ dominating\ factor\ therefore\ T(n) = O(n\, log\, n)$$

```
/*
 * ADSA Assignment 2, question 3
 *
 * a1077337 Xiang Guo
 * a1646630 Patrick Man
 *
 */

import java.rmi.RemoteException;

public class PolyMulti {

        /**
         * @param args
         */

        public static int[] polyMulti(int[] p, int[] q)throws RemoteException{
                int n = p.length;
                int m = q.length;

                /*
                 * Base case
                 */

                if (n <= 1 && m <=1){
                        int[] r = new int[1];
                        r[0] = multiply(p[0],q[0]);
                        return r;
                }

                /*
                 * Split p into p1 and p2
                 */
                int p1_len= n/2;
                int[] p1 = new int[p1_len];
                int[] p2 = new int[n - p1_len];

                if(n > 1){
                        int k = n/2;
                        for (int i = 0; i < k; i++)
                                p1[i] = p[i];

                        for (int i = 0; i < (n-k); i++)
                                p2[i] = p[k+i];
                }

                /*
                 * Split q into q1 and q2
                 */
                int q1_len = m/2;
                int[] q1 = new int[q1_len];
                int[] q2 = new int[m - q1_len];

                if(m > 1){
                        int l = m/2;
                        for (int i = 0; i < l; i++)
```

```
                                q1[i] = q[i];

                        for (int i = 0; i < (m-l); i++)
                                q2[i] = q[l+i];
                }


                /*
                 * Recursive call
                 */
                int A[] = polyMulti(p1,q1);
                int C[] = polyMulti(p2,q2);

                /*
                 * Calculate (p1(x) + p2(x))
                 */
                int[] p3 = new int[max(p1.length,p2.length)];

                for(int i = 0; i < q1.length; i++){
                        p3[i] = p1[i] + p2[i];
                        if(p1.length<p2.length){
                                p3[i+1] = 0 + p2[i+1];
                        }
                }

                /*
                 * Calculate (q1(x) + q2(x))
                 */
                int[] q3 = new int[max(q1.length,q2.length)];

                for(int i = 0; i < q1.length; i++){
                        q3[i] = q1[i] + q2[i];
                        if(q1.length<q2.length)
                                q3[i+1] = 0 + q2[i+1];
                }

                /*
                 * Call B(x) = ((p1(x) + p2(x)) x ((q1(x) + q2(x)) - A(x) - C(x)
                 *
                 */
                int B[] = new int[p1.length+p2.length];
                B =     polyMulti(p3,q3);

                for( int i = 0; i < A.length ; i++){
                        B[i] = B[i] - A[i];
                }

                for( int i = 0; i < C.length ; i++){
                        B[i] = B[i] - C[i];
                }


                /*
                 * Put A B and C into array r
                 *
                 */
                int[] r = new int[n*2-1];
                for(int i = 0; i < A.length; i++){
                        r[i] = r[i] + A[i];
                }

                for(int i = 0; i < B.length; i++){
                        r[i+n/2] = r[i+n/2] + B[i];
                }

                for(int i = 0; i < C.length; i++){
                        r[i+(n/2)*2] = r[i+(n/2)*2] + C[i];
                }

                return r;
        }

        public static int multiply(int a, int b){
                int c = a * b;
                return c;
        }

        public static int max(int a, int b){
                if(a > b)
                        return a;
                else
                        return b;

        }
}
```

Question 3.3

a)

```
 p[] = 5 0 -6 8 3 -9 0 8
 q[] = 0 10 -9 8 -3 0 0 9
 result[] = 0 50 -45 -20 119 -90 -35 126 -1 -99 136 3 -81 0 72
```

 b)

```
 p[] = -9 -7 0 0 7 9 8 -7 0 -3 0 8 0 -3 0 2
 q[] = 0 0 8 -2 0 3 -5 4 0 -6 1 -9 6 3 8 0
 result[] = 0 0 -72 -38 14 -27 80 57 18 3 39 57 -10 20 -193 -167 29 -28 231 66 16
-76 -55 21 39 48 -27 -12 6 16 0
```