************ OUTPUT ***********************

These 6 scenarios will be tested:
Also note that I have not considered k at all yet.
This will be implemented a bit later.
Also not taken opportunity to rebuy at:
int[] stock = new int[]{5,2,4,0,1};

int[] stock = new int[]{1,3,2,8,4,10};
int[] stock = new int[]{5,2,4,0,1};
int[] stock = new int[]{1,1,8,12,15};
int[] stock = new int[]{1,3,2,8,10};
int[] stock = new int[]{5,4,8};
int[] stock = new int[]{0,4,6,3,2};

This will exclude buying at 0 and selling at profit.

int[] stock = new int[]{1,3,2,8,10};
For instance…. In following:
It would be best to buy at 1 and sell at 10 (this would give maximum 9)
However this is not how the coding problem described the scenario. It suggested:
buying and selling on the basis that it can be bought at a lower selling point.
In which case, it processed….. 3-1 + 8-2 = TOTAL 8
The exception to the above rule is if there is no point in selling with exception of the last element. Such as this sequence:
int[] stock = new int[]{1,1,8,12,15};

*** OUTPUT (FULL SCREEN OUTPUT) *** FOR OTHER STOCKS ABOVE, INFORMATION WILL BE REDUCED

```
Welcome to Online IDE!! Happy Coding :)

Stocks bought
[1, 3, 2, 8, 4, 10]

*************************

Stock being evaluated: 1
```

```
*************************
1
Next highest stock is:3 after 1
this is the profit so far1: 2(3-1)
This is the RUNNING SCENARIO TOTAL: 2
This is the next highest stock: 8 after 2
this is the profit so far3: 8(8-2+2)
This is the RUNNING SCENARIO TOTAL: 8
this is the stock: 1
this is the num: 0
WHY NOT HERE!!!!!!!!!!!
3
1
value of j:1
length:5
WHY NOT HERE!!!!!!!!!!!
2
1
value of j:2
length:5
WHY NOT HERE!!!!!!!!!!!
8
1
value of j:3
length:5
ENTERRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
Next highest stock is:10 after 4
this is the profit so far1: 14(10-4+8)
This is the RUNNING SCENARIO TOTAL: 14
WHY NOT HERE!!!!!!!!!!!
4
1
value of j:4
length:5
WHY NOT HERE!!!!!!!!!!!
10
1
value of j:5
length:5
This is the stock:1
This is all possible routes to make profit:6
VALUE HERE:

Stocks bought
[1, 3, 2, 8, 4, 10]

*************************

Stock being evaluated: 3
*************************
3
WHY NOT HERE!!!!!!!!!!!
2
3
value of j:2
length:5
WHY NOT HERE!!!!!!!!!!!
8
3
value of j:3
length:5
ENTERRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
Next highest stock is:10 after 4
this is the profit so far1: 6(10-4+0)
This is the RUNNING SCENARIO TOTAL: 6
```

```
WHY NOT HERE!!!!!!!!!!!
4
3
value of j:4
length:5
WHY NOT HERE!!!!!!!!!!!
10
3
value of j:5
length:5
This is the stock:3
This is all possible routes to make profit:6
VALUE HERE:

Stocks bought
[1, 3, 2, 8, 4, 10]

*************************

Stock being evaluated: 2
*************************
2
WHY NOT HERE!!!!!!!!!!!
8
2
value of j:3
length:5
ENTERRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
Next highest stock is:10 after 4
this is the profit so far1: 6(10-4+0)
This is the RUNNING SCENARIO TOTAL: 6
WHY NOT HERE!!!!!!!!!!!
4
2
value of j:4
length:5
WHY NOT HERE!!!!!!!!!!!
10
2
value of j:5
length:5
This is the stock:2
This is all possible routes to make profit:6
VALUE HERE:

Stocks bought
[1, 3, 2, 8, 4, 10]

*************************

Stock being evaluated: 8
*************************
8
WHY NOT HERE!!!!!!!!!!!
4
8
value of j:4
length:5
WHY NOT HERE!!!!!!!!!!!
10
8
value of j:5
length:5
This is the stock:8
This is all possible routes to make profit:0
```

```
VALUE HERE:

Stocks bought
[1, 3, 2, 8, 4, 10]

*************************

Stock being evaluated: 4
*************************
4
WHY NOT HERE!!!!!!!!!!!
10
4
value of j:5
length:5
This is the stock:4
This is all possible routes to make profit:0
VALUE HERE:

Stocks bought
[1, 3, 2, 8, 4, 10]

*************************

Stock being evaluated: 10
*************************
10
This is the stock:10
This is all possible routes to make profit:0
VALUE HERE:
ALL VALUES
8
ALL VALUES
14
ALL VALUES
6
ALL VALUES
6
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
ALL VALUES
```

```
0
ALL VALUES
0
ALL VALUES
0
This is the highest profit possible: 14


** Process exited - Return Code: 0 **
```

Welcome to Online IDE!! Happy Coding :)

Stocks bought
[5, 2, 4, 0, 1]

```
*************************

Stock being evaluated: 5
*************************
5
WHY NOT HERE!!!!!!!!!!!
2
5
value of j:1
length:4
WHY NOT HERE!!!!!!!!!!!
4
5
value of j:2
length:4
WHY NOT HERE!!!!!!!!!!!
0
5
value of j:3
length:4
WHY NOT HERE!!!!!!!!!!!
1
5
value of j:4
length:4
This is the stock:5
This is all possible routes to make profit:0
VALUE HERE:

Stocks bought
[5, 2, 4, 0, 1]

*************************

Stock being evaluated: 2
*************************
2
Next highest stock is:4 after 2
this is the profit so far1: 2(4-2)
This is the RUNNING SCENARIO TOTAL: 2

*************************

Stock being evaluated: 4
*************************
4
WHY NOT HERE!!!!!!!!!!!
```

```
0
4
value of j:3
length:4
WHY NOT HERE!!!!!!!!!!!
1
4
value of j:4
length:4
This is the stock:4
This is all possible routes to make profit:0
VALUE HERE:

Stocks bought
[5, 2, 4, 0, 1]

*************************

Stock being evaluated: 0
*************************
0
WHY NOT HERE!!!!!!!!!!!
1
0
value of j:4
length:4
This is the stock:0
This is all possible routes to make profit:0
VALUE HERE:

Stocks bought
[5, 2, 4, 0, 1]

*************************

Stock being evaluated: 1
*************************
1
This is the stock:1
This is all possible routes to make profit:0
VALUE HERE:
ALL VALUES
0
ALL VALUES
2
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
```

```
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
This is the highest profit possible: 2

** Process exited - Return Code: 0 **
```

```
Welcome to Online IDE!! Happy Coding :)

Stocks bought
[1, 1, 8, 12, 15]

*************************

Stock being evaluated: 1
*************************
Next highest stock is:15 after 12
this is the profit so far1: 3(15-12+0)
This is the RUNNING SCENARIO TOTAL: 3

Stocks bought
[1, 1, 8, 12, 15]

*************************

Stock being evaluated: 1
*************************
Next highest stock is:15 after 12
this is the profit so far1: 3(15-12+0)
This is the RUNNING SCENARIO TOTAL: 3

Stocks bought
[1, 1, 8, 12, 15]

*************************

Stock being evaluated: 8
*************************
Next highest stock is:15 after 12
this is the profit so far1: 3(15-12+0)
```

```
This is the RUNNING SCENARIO TOTAL: 3

*************************

Stock being evaluated: 12
*************************

*************************

Stock being evaluated: 15
*************************
15
This is the stock:15
This is all possible routes to make profit:0
VALUE HERE:
ALL VALUES
3
ALL VALUES
3
ALL VALUES
3
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
This is the highest profit possible: 3


** Process exited - Return Code: 0 **
```

Welcome to Online IDE!! Happy Coding :)

Stocks bought
[1, 3, 2, 8, 10]

```
*************************

Stock being evaluated: 1
*************************
1
Next highest stock is:3 after 1
this is the profit so far1: 2(3-1)
This is the RUNNING SCENARIO TOTAL: 2
this is the profit so far2: 4(10-8+2)
This is the RUNNING SCENARIO TOTAL: 4
ENTERRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
Next highest stock is:10 after 8
this is the profit so far1: 6(10-8+4)
This is the RUNNING SCENARIO TOTAL: 6

*************************

Stock being evaluated: 3
*************************
Next highest stock is:10 after 8
this is the profit so far1: 2(10-8+0)
This is the RUNNING SCENARIO TOTAL: 2


*************************
Stock being evaluated: 2
*************************
2
ENTERRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
Next highest stock is:10 after 8
this is the profit so far1: 2(10-8+0)
This is the RUNNING SCENARIO TOTAL: 2

*************************

Stock being evaluated: 8
*************************



*************************

Stock being evaluated: 10
*************************
10
This is the stock:10
This is all possible routes to make profit:0
VALUE HERE:
ALL VALUES
4
ALL VALUES
6
ALL VALUES
2
ALL VALUES
2
ALL VALUES
0
ALL VALUES
0
ALL VALUES
```

```
0
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
ALL VALUES
0
This is the highest profit possible: 6


** Process exited - Return Code: 0 **
```

```
Welcome to Online IDE!! Happy Coding :)
```

Stocks bought
[5, 4, 8]

```
*************************

Stock being evaluated: 5
*************************
This should be last stock:8
This is the RUNNING SCENARIO TOTAL: 3
this is the profit so far55: 3(8-5)

*************************

Stock being evaluated: 4
*************************
This should be last stock:8
This is the RUNNING SCENARIO TOTAL: 4
this is the profit so far55: 4(8-4)

*************************

Stock being evaluated: 8
*************************
This is the highest profit possible: 4
```

```
** Process exited - Return Code: 0 **

---------------------------------------------------------------------------------------------------------------------

Welcome to Online IDE!! Happy Coding :)

Stocks bought
[0, 4, 6, 3, 2]

*************************

Stock being evaluated: 0
*************************

*************************

Stock being evaluated: 4
*************************
Next highest stock is:6 after 4
this is the profit so far1: 2(6-4)
This is the RUNNING SCENARIO TOTAL: 2




*************************

Stock being evaluated: 6
*************************

*************************

Stock being evaluated: 3
*************************

*************************

Stock being evaluated: 2
*************************
This is the highest profit possible: 2


** Process exited - Return Code: 0 **
```

*** CODE ***

```
/*
Online Java - IDE, Code Editor, Compiler
Online Java is a quick and easy tool that helps you to build, compile, test your programs online.
*/
import java.io.*;
import java.util.*;
public class Main
{
  public static void main(String[] args) {
    System.out.println("Welcome to Online IDE!! Happy Coding :)");
    //According to this problem you can buy the stock at 0 and sell it for a profit.
    // For example the solution to this problem states
    // for stocks    5,2,4,0,1     (4-2)  +  (1-0)   =  3
    // However for this code I have prevented this.
    // Here are some scenarios created to test this problem. All seem to function.
    // the value of k has not been implemented

     //int[] stock = new int[]{1,3,2,8,4,10};  //CORRECT     (3-1) + (8-2) + (10-4) = 14
     //int[] stock = new int[]{5,2,4,0,1};  // CORRECT    (4-2)  = 2
     //int[] stock = new int[]{1,1,8,12,15};   // CORRECT (15-12) = 3
     //int[] stock = new int[]{1,3,2,8,10};  //PASSES (3-1) +  (10-8) = 4
     //int[] stock = new int[]{5,4,8};  //PASSES  (8-4)
    //int[] stock = new int[]{0,4,6,3,2};  //PASSES (6-4)
     //int[] stock = new int[]{5,9,8,2};  // PASSES (9-5)
     //int[] stock = new int[]{1,2,3,4,5,6,7,8};  // PASSES
    //int[] stock = new int[]{19,5,32,7,15};  //PASSES  (32-5) + (15-7) = 35




    //EXTRA TESTS
    //int[] stock = new int[]{5,2,15,4,18,24,9,1,1,2};  //FAILS   IT BUYS (18-4) even though 24 is higher than
18
    //int[] stock = new int[]{0,1,0,1,0,2,0,1,1,1};  // PASSES   0
    //int[] stock = new int[]{16,1,8,12,17};   // CORRECT
    //int[] stock = new int[]{1,0,9,7,5,2,6,3};   // PASSES  (9-1)

    int[][] difference = new int[stock.length][20];
    int count=0;
    int temp=0;
    int k;
    boolean profitPossible=false;

    int tempStore=0;
    int runningTotal=0;
    int num=20;
    int [] profit = new int[num];
    num=0;
    int max=0;
    int penultimateProfit=0;
    boolean previousProfit=false;
    boolean bypassPenultimate=false;

    for (int i=0; i<stock.length;i++)
    {
      System.out.println("\nStocks bought");
      System.out.println(Arrays.toString(stock));
      System.out.println("\n**************************");
      System.out.println("\nStock being evaluated: " + stock[i]);
```

```
        System.out.println("**************************");


        //System.out.println("current running total:" + runningTotal);

        System.out.println(stock[i]);
        int used=9999999;
        count=0;
        runningTotal=0;
        //penultimateProfit=false;
        previousProfit=false;
        bypassPenultimate=false;
        //penultimateProfit=false;

        for (int j=i+1; j<stock.length;j++)  // this ensures stock is not compared against itself
        {
           if (j!=stock.length-1)
           {
              if (j==i)
           {
              j++;
           }

            // this checks if next stock is greater or higher than examined stock
            // and also if one after next is lower. These conditions will allow point of sale and rebuy at new
value
            // this is to try and fix circumstance for 1, 3 ,2, 8, 10
            //currently it is doing 10 - 8 and skipping 10-2
            // this will try to accomodate for this condition.

              if (stock[j]>=stock[i]&& stock[j+1]>stock[j] && stock[i]!=0 && j+1==stock.length-1 &&
!bypassPenultimate /*&& used!=stock[j]*/)
           {



              // this is to ensure that in stocks as below, it does not process.... 10-8 on two instances
              //int[] stock = new int[]{1,3,2,8,10};

              if (temp==stock.length-1 || temp+1 == stock.length-1)
             {
                System.out.println("GETTTTT OUT!!!!");
                break;
             }
              System.out.println("ENTERRRRRRRRRRRRRRRRRRRRRRRRRRRRRR");
              profitPossible=true;
              System.out.println("Next highest stock is:" + stock[j+1] + " after " + stock[j]);
              tempStore=difference[i][count];
              difference[i][count]=stock[j+1]-stock[j];
              runningTotal=runningTotal + difference[i][count];

             //System.out.println("this is the profit so far1: " + difference[i][count] + "("+stock[j+1] + "-" +
stock[j]+")");
             System.out.println("this is the profit so far1: " + runningTotal + "("+stock[j+1] + "-" + stock[j] + "+"
+  tempStore+")");

             profit[num]=runningTotal;
             previousProfit=true;
             System.out.println("This is the RUNNING SCENARIO TOTAL: " + profit[num]);
             bypassPenultimate=true;
           }
           System.out.println("This is used: " + used);
```

```
            System.out.println("This is i: " + i);
            //used!=i   is used since it prevents
            //5, 9 , 8 ,2   (9-5) and (8-5)



            if (stock[j]>=stock[i]&& stock[j+1]<stock[j] && stock[i]!=0 && temp!=j && !bypassPenultimate &&
used!=i && stock[j+1]!=0)
            {
              profitPossible=true;
             System.out.println("USSSSSSSSSSSSEDDD value: " + used);
              System.out.println("Next highest stock is:" + stock[j] + " after " + stock[i]);
              difference[i][count]=stock[j]-stock[i];
              runningTotal=runningTotal + difference[i][count];
              System.out.println("this is the profit so far123: " + difference[i][count] + "("+stock[j] + "-" +
stock[i]+")");

             profit[num]=runningTotal;
             used=i;
             System.out.println("USED VALUEEEEEEE: " + used);

             previousProfit=true;
             System.out.println("This is the RUNNING SCENARIO TOTAL: " + profit[num]);
              // this will check all other occurrences from the current point if there is scope to sell and buy
              //conditions are similar
              for (temp=j+1;temp<stock.length;temp++)
              {
                System.out.println("What is stock j:" + stock[j]);
                System.out.println("What is stock i:" + stock[i]);
                System.out.println("What is stock temp:" + stock[temp]);
                // conditions are similar.. if next item is same or greater.. And following stock is lower than
previous
                if (stock[temp]>=stock[j] && stock[j+1]<stock[j] && !bypassPenultimate && stock[temp-1]!=0
&& stock[j+1]!=0)
                {
                  if (temp+1==stock.length-1)
// this is checking if last stock in list and to avoid any exceptions, it breaks out of loop
                  {
                    tempStore=difference[i][count];
                    difference[i][count]=difference[i][count] + (stock[temp+1]-stock[temp]);
                     if (previousProfit)
                    {
                      runningTotal=difference[i][count];
                    }
                    if (!previousProfit)
                    {
                      runningTotal=runningTotal + difference[i][count];
                    }
                    //runningTotal=runningTotal + difference[i][count];
                    System.out.println("this is the profit so far2: " + difference[i][count] + "("+ stock[temp+1]
+ "-" + stock[temp] +"+"+ tempStore + ")");

                    System.out.println("this is the stock: " + stock[i]);
                    System.out.println("this is the num: " + num);
                    profit[num]=runningTotal;
                     System.out.println("This is the RUNNING SCENARIO TOTAL: " + profit[num]);
                    num++;
                    break;
                  }
                  System.out.println("This is the next highest stock: " + stock[temp] + " after " + stock[temp-
1]);
```

```
                  tempStore=difference[i][count];
                  difference[i][count]=difference[i][count] + (stock[temp]-stock[temp-1]);
                  System.out.println("this is the profit so far3: " + difference[i][count] + "("+ stock[temp] + "-"
+ stock[temp-1] +"+"+ tempStore + ")");
                  //positionCompared=temp;

                  if (previousProfit)
                    {
                      runningTotal=difference[i][count];
                    }
                    if (!previousProfit)
                    {
                      runningTotal=runningTotal + difference[i][count];
                    }

                  //runningTotal=runningTotal + difference[i][count];
                  profit[num]=runningTotal;
                  //System.
                 System.out.println("This is the RUNNING SCENARIO TOTAL: " + profit[num]);
                  System.out.println("this is the stock: " + stock[i]);
                  System.out.println("this is the num: " + num);

                  num++;
                  break;
                }

              }
            }

            }

            // this is now creating a scenario to examine if there has been a buy and resell
            // it also checks scenario such as  2,4,0,1  since the current logic above
            // does not account for a buy and sell as last two stocks

            if (stock[j]>stock[j-1] && profitPossible && j==stock.length-1 && i==stock.length-2 &&
bypassPenultimate)
            {
              System.out.println("This should be penultimate stock:" + stock[i]);
              System.out.println("This should be last stock:" + stock[j]);

              tempStore=difference[i][count];
              runningTotal = runningTotal + (stock[j]-stock[j-1]);

              // It is capturing value here
              penultimateProfit=(stock[j]-stock[j-1]);
              num++;
              profit[num]=runningTotal;

               System.out.println("This is the RUNNING SCENARIO TOTAL: " + profit[num]);
              System.out.println("this is the stock: " + stock[i]);
              System.out.println("this is the num: " + num);
              System.out.println("this is the profit so far44: " + runningTotal + "("+stock[j] +"-"+ stock[j-1]+")");
              profit[num]=runningTotal;
              System.out.println("************REACH********");
              break;
            }

            // this now covers scenario such as   1 , 1, 8 , 15
            // without this logic, there will be no buy sell and buy
            // checks if stock is higher than examined stock.
            // it will only enter this scenario also if no other profits have been analysed
            // otherwise totals will be incorect.. For instance, it would process  8 => 15 when it
            // would be completed as part of normal logic.
            System.out.println("WHY NOT HERE!!!!!!!!!!!");
            System.out.println(stock[j]);
            System.out.println(stock[i]);
            System.out.println("value of j:" + j);
            System.out.println("length:" + (stock.length-1));
             if (stock[j]>stock[i] && !profitPossible && j==stock.length-1 && stock[i]!=0)
            {
              //System.out.println("Rare instance of all increasing");
              System.out.println("This stock:" + stock[i]);
              System.out.println("This should be last stock:" + stock[j]);

              tempStore=difference[i][count];
              if (previousProfit)
                    {
                      runningTotal = runningTotal + (stock[j]-stock[i]);
                    }
                    if (!previousProfit)
                    {
                      runningTotal=(stock[j]-stock[i]);
                    }
              //runningTotal = runningTotal + (stock[j]-stock[i]);
              profit[num]=runningTotal;
              System.out.println("This is the RUNNING SCENARIO TOTAL: " + profit[num]);
               num++;
              System.out.println("this is the profit so far55: " + runningTotal + "("+stock[j] +"-"+ stock[i]+")");
              System.out.println("************REACH AGAIN********");
              break;
            }
        }
        System.out.println("This is the stock:" + stock[i]);
              System.out.println("This is all possible routes to make profit:" + difference[i][count]);
              //profit[num]=runningTotal;
              System.out.println("VALUE HERE:");
             //System.out.println(profit[num]);
 num++;
 count++;
   }
     // this is simply assigning the highest profitable route
   for (int p=0;p<profit.length;p++)
{
   System.out.println("ALL VALUES");
   System.out.println(profit[p]);
   if (profit[p]>max)
   {
     max=profit[p];
   }
}
// this gets the profit total if shares are not as such:
//5,2,4,0,1
// this now checks if scenario such as above
if (penultimateProfit!=0 && previousProfit)
{
System.out.println("This is the highest profit possible1: " + (max + penultimateProfit));
System.out.println(max);
System.out.println(penultimateProfit);
}
else
{
```

```java
    System.out.println("This is the highest profit possible: " + max);
}
}
}
```