(1) Given a list of stock prices ordered by time for a single stock, find the difference between the best price to purchase and the best price to then sell the stock.

First, the divide-and-conquer solution. This can be solved by splitting the input in two array and solving the problem in each subarray, then combining the two together.

Algorithm :

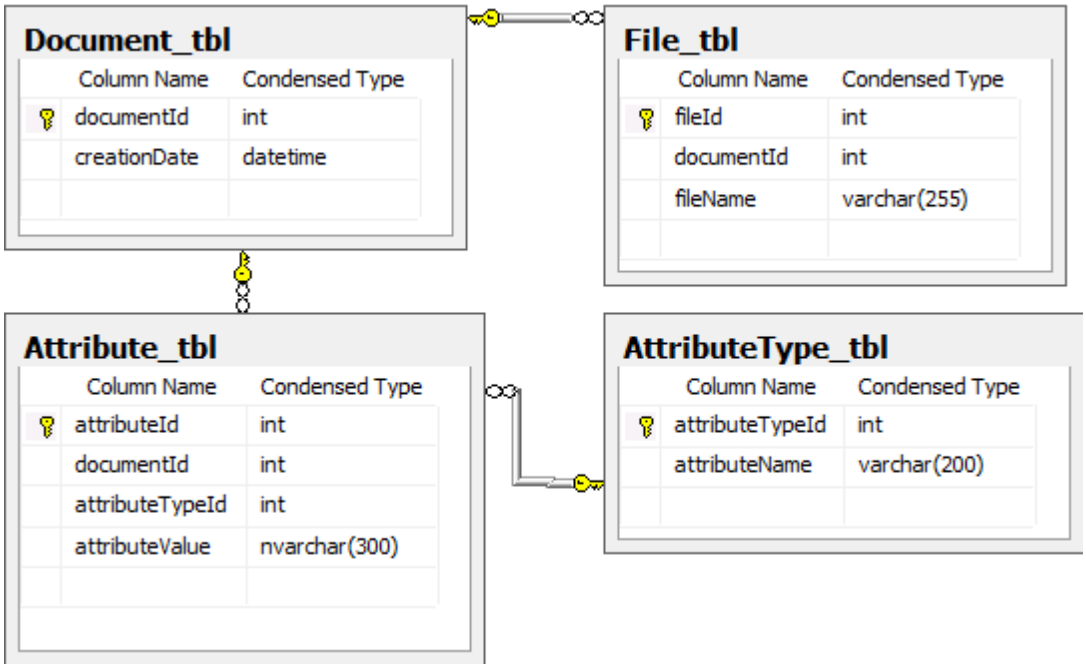
```
var max_profit = 0;
var stockPrices = [23,40,21,67,1,50,22,38,2,62];

var currentBestBuy = 0;
var currentBestSell = 0;
var min = 0;

for(var i = 0;i < (stockPrices.length - 1) ; i++){
    if(( stockPrices[i + 1] - stockPrices[currentBestBuy] > max_profit) ){
        max_profit = stockPrices[i + 1] - stockPrices[currentBestBuy];
        currentBestSell = i + 1;
    }
    if(stockPrices[i] < stockPrices[currentBestBuy]){
            min = i;
        }
    if( max_profit < stockPrices[i + 1] - stockPrices[min] ){
        max_profit = stockPrices[i + 1] - stockPrices[min];
        currentBestSell = i + 1;
        currentBestBuy = min;
    }
}
//  the best price to purchase
console.log(currentBestBuy);
// best price to then sell the stock
console.log(currentBestSell);
// difference or Profit
console.log(max_profit);
```

(2) Given two arrays, A1 and A2, find all the elements in A2 that are not in A1. Comment on the algorithmic complexity of your solution.

```
using System;
namespace MissingElement
{
    class Program
    {
        static void FindMissingInSecondArray (int[] a, int[] b,int n, int m)
        {
            for (int i = 0; i < m; i++)
            {   int j;
                for (j = 0; j < n; j++)
                    if (b[i] == a[j])
                        break;
                if (j == n)
                    Console.Write(b[i] + " ");
            }
        }
        static void Main(string[] args)
        {
            int[] A1 = { 1, 2, 6, 3, 4, 5 };
            int[] A2 = { 2, 4, 3, 1, 0 };
            int n = A1.Length;
            int m = A2.Length;
            FindMissingInSecondArray (A1, A2, n, m);
            Console.ReadKey();
        }
    }
}
```

Complexity: A Naive Approach is to use two loops and check element which not present in first array.

(4) Given the below data model and sample data:



AttributeType_tbl	
attributeTypeId	**attributeName**
1	filingDate
2	formType
3	filingCompany

Document_tbl	
documentId	**creationDate**
2	3/30/2013
6	3/6/1987
11	2/20/2012

File_tbl		
fileId	**documentId**	**fileName**
1	2	OKTO2.pdf
4	6	"Mainsail" Liquid Engine.xlsx
10	11	"Poodle" Liquid Engine.docx

Attribute_tbl			
attributeId	**documentId**	**attributeTypeId**	**attributeValue**
1	6	2	Marketing Materials
2	6	2	Spec Sheet
3	6	1	3/31/1987
4	6	3	Rockomax
5	2	1	1/1/2013
6	2	2	Marketing Materials
7	2	3	Probodobodyne
8	11	2	Marketing Materials
9	11	1	1/2/2012
10	11	3	Rockomax

(4a) Find all documents filed by Rockomax that are of formType “Marketing Materials” and not formType “Spec Sheet”. Return the documentId, creationDate, and formType .

i. Before you actually write the query, based on the above sample data, which document(s) will be returned (just write the documentId).

ii. Write the query.

Answer to the Q no 4a part (i.)

documentId returned will be 6 & 11.

Answer to the Q no 4a part (ii.)

```
Select TableA.documentId,creationDate,formType from (
select DT.documentId ,creationDate,attributeValue as formType from Document_tbl DT inner
join Attribute_tbl AT
on DT.documentId=AT.documentId inner join AttributeType_tbl ATT on
AT.attributeTypeId=ATT.attributeTypeId
inner join File_tbl FT on DT.documentId=FT.documentId
Where attributeValue='Marketing Materials' ) TableA

inner join (

select Document_tbl.documentId,attributeValue as FiledBy from Document_tbl inner join
File_tbl on Document_tbl.documentId=File_tbl.documentId
inner join Attribute_tbl on Document_tbl.documentId=Attribute_tbl.documentId where
attributeValue='Rockomax'
) TableB on TableA.documentId=TableB.documentId
```

(5) Write a function that is capable of adding two very large positive numbers (~10^{100}) together, given as strings.

```
Public class AddTwoLargerNumber
    {

        static string findSum(string str1, string str2)
        {
            // Before proceeding further, make sure length
            // of str2 is larger.
            if (str1.Length > str2.Length)
            {
                string t = str1;
                str1 = str2;
                str2 = t;
            }
            string str = "";

            int n1 = str1.Length, n2 = str2.Length;

            // Reverse both of strings
            char[] ch = str1.ToCharArray();
            Array.Reverse(ch);
            str1 = new string(ch);
            char[] ch1 = str2.ToCharArray();
            Array.Reverse(ch1);
            str2 = new string(ch1);

            int carry = 0;
            for (int i = 0; i < n1; i++)
            {

                int sum = ((int)(str1[i] - '0') +
                        (int)(str2[i] - '0') + carry);
                str += (char)(sum % 10 + '0');
                carry = sum / 10;
            }

            for (int i = n1; i < n2; i++)
            {
                int sum = ((int)(str2[i] - '0') + carry);
                str += (char)(sum % 10 + '0');
                carry = sum / 10;
            }

            if (carry > 0)
                str += (char)(carry + '0');

            // reverse resultant string
            char[] ch2 = str.ToCharArray();
            Array.Reverse(ch2);
            str = new string(ch2);
            return str;
        }
        static void Main(string[] args)
        {
            string str1 = "12";
            string str2 = "198111";
        }
               Console.WriteLine(findSum(str1, str2));
        }
```