Project in ME001 – Sampling system Group 1

By Chen YuXuan 1809853J-I011-0011 D1

Wang Yuan 1809853G-I011-0030 D1

He PeiLin 1809853U-I011-0078 D1

December 2, 2020

Contents

1 Restatement of the Problem

In this project, we are expected to extract a subset of samples of big data. Assume there are m samples ($45 \leq m \leq 54$), any n ($7 \leq n \leq 25$) samples out of these m samples are selected. There are C_n^m groups of n samples. From one of these groups of n samples, we randomly selected k ($4 \leq k \leq 7$) samples to form some groups. So there will be C_n^k groups of k samples selected. There are at least **ONE** group of k samples, in which s ($3 \leq s \leq 7$) samples have been selected from the j (where $s \leq j \leq k$) samples. Among these groups of k samples, we would like to optimize them by selecting ONLY some of them.

2 Basic Ideas

We can divide the problem into two parts, $j = s$ and $j \neq s$.

2.1 $j = s$

2.1.1 Algorithm to Find Subsets

Now, we have a set whose number of the element is n. Then we want to find out all the subsets whose number of the element is k.

Algorithm:

- First, we put the origin set to a container, and then we label every element to one(illustrate the picture below). We assume that the origin set is S, $S = \{1, 2, 3, 4, 5\}$ in Table.2.1.1. Then, the subset which has the same element with the original set's is labeled the element to 1, otherwise labeling it to 0. For example, we suppose that one the subset is S_1,$S_1 = \{1, 2, 4\}$. We can represent it as Table.2.1.2. Now we can change the number below the array to a binary number, which means that each subset can be represented by a unique number from 0(empty set) to $2^n - 1$(original set). Just like the example above set S can be represented by $11111_2 = 31_{10}$ and S_1 can be expressed as $01011_2 = 11_{10}$

5	4	3	2	1
1	1	1	1	1

Table. 2.1.1

5	4	3	2	1
0	1	0	1	1

Table. 2.1.2

- Subsequently, we know how to find subsets of the original set, but I want to know how to find the subset with the specific number of elements. Therefore, we only need to know the subset whose binary number representation contains k 1s. As the example in Table.2.1.2, $S_1 = \{1, 2, 4\}$: So, the S_1 contains three elements, because it has three *1*s.

  In this way, we can easily find out the subset whose number of elements is k from 0 to $2^n - 1$, the code block findSubsetOfk illustrates the situation.

```
void findSubsetOfk(int n,int k, vector<int> subsetK){
    int count=0;//number of 1s
    for(int i = 1 ; i < (1<<n); i++){
        for(int j = 0; j < n; j++){
            //the binary number representation
            //of subset has an 1 on the jth position
            if(i & (1<<j)!=0){
                count++;
            }
        }
        if(count==k)
            susetK.empalce_back(i);
        count=0;
    }

}
```

  However, we can easily find that the binary number representation of the subset whose number of elements is k is no less than $2^k - 1$. Therefore, we the code above, we can have an optimization on the i. The optimized code findSubsetOfkOptim is

```
void findSubsetOfkOptim(int n,int k, vector<int> subsetK){
    int count=0;//number of 1s
    for(int i = (1<<k)-1 ; i < (1<<n); i++){
        for(int j = 0; j < n; j++){
            //the binary number representation
            //of subset has an 1 on the jth position
            if(i & (1<<j)!=0){
                count++;
            }
        }
        if(count==k)
            susetK.empalce_back(i);
        count=0;
    }

}
```

- Currently, we can use the same way what we say above to find out the subset of the set whose number of element is k and its number of elements is s.

2.1.2 Calculate the Combination Number

If we calculate the combination number directly, it is likely to out of bounds of int. So we can use combination formula:

$$C_n^m = C_{n-1}^{m-1} + C_{n-1}^m$$

to calculate the combination number. And the specific implementation code can be seen in calculateCombination.

```
int calculateCombinationNumber(int n,int m){
    for(int i=0;i<=n;i++)
        C[i][0]=1;
    for(int i=1;i<=n;i++)
        for(int j=1;j<=i;j++)
            C[i][j]=C[i-1][j-1]+C[i-1][j];
    return C[n][m];
}
```

2.1.3 Greedy Algorithm to Calculate the Set Coveraged

We denote that the input is a set $\mathcal{U}$ of n elements, and a collection $S = \{S_1, S_2, ..., S_m\}$ of m subsets of $\mathcal{U}$ such that $\cup_i S_i = \mathcal{U}$. Our goal is to take as few subsets as possible from S such that their union covers $\mathcal{U}$. We can solve this problem easily by greedy algorithm. The algorithm is below in Table.2.1.3:

```
Greedy Cover($S$,$\mathcal{U}$)
repeat
    pick the set that covers the maximum number of uncover element
    mark elements in the chosen set as covered
    remove the set from $S$ to the result set
done
```

Table. 2.1.3. Greedy Cover

Based on the three lemmas above, we can easily transform the problem to that the set $\mathcal{U} = \{1, 2, \cdots, C_n^j\}$, which means that we map each different subset whose the number of the elements is j to a unique code from 1 to C_n^j. Each subset of S, represents the each k set's subsets whose number of elements is j. Ultimately, we can solve the problem easily.

2.2 $j \neq s$

The way to solve the problem is just like the way we mentioned above. However, after finishing finding the subset of the k set whose element number is s, we should know how many sets whose the number of elements is j include it. Therefore, we use **DFS(depth first search)** to find out them. Assuming that $n = 5, s = 3, j = 4$, and the subset whose number of elements is equal to 3 is labeled as 01011_2. Therefore, we can expand it as below in Table.2.2.1.

5	4	3	2	1
0	1	0	1	1
0	1	1	1	1
1	1	0	1	1

Table. 2.2.1

Then, we should mark the last two rows of the set above in the $\mathcal{U}$ as covered.

3 Essential Codes and Functions Analysis

3.1 Realization of Modifying DB files

As the request said, we need output the group of k samples and corresponding result in DB files. Consequently, the OOP program language **C#** can provide abundant libraries to help realize combine with modifying DB files.

Depending on **C#** powerful library and interface, we can apply our algorithm source code on GUI platform, and realize the operation of creating new files(Code.1), exporting result into corresponding files(Code.2) as well as deleting the specular data(Code.3).

```
public void CreateTableInToMdb(string fileNameWithPath)
{
    try
    {
        OleDbConnection myConnection = new OleDbConnection
            ("Provider=Microsoft.Jet.OLEDB.4.0; Data Source="
                + fileNameWithPath);
        myConnection.Open();
        OleDbCommand myCommand = new OleDbCommand();
        myCommand.Connection = myConnection;
        myCommand.CommandText =
        "CREATE TABLE my_table([m] NUMBER, " +
            "[n] NUMBER, [k] NUMBER, [j] Number, " +
                "[s] NUMBER, [n numbers] TEXT, " +
                    "[minium number of sets] NUMBER, "+
                        "[answer] TEXT)";
        myCommand.ExecuteNonQuery();
        myCommand.Connection.Close();
    }
    catch { }
}
```

```
public void InsertToMdb(string fileNameWithPath)
{
    var con = new OleDbConnection(
        "Provider = Microsoft.Jet.OLEDB.4.0; Data Source = "
            + fileNameWithPath);
    var cmd = new OleDbCommand();
    cmd.Connection = con;
    cmd.CommandText = "insert into my_table ([m],[n],[k],[j]," +
        "[s],[n numbers],[minium number of sets], [answer])" +
            "values (@m, @n, @k,@j,@s,@series1 , @number, @answer);";
    cmd.Parameters.AddWithValue("@m", numericUpDown1.Value);
    cmd.Parameters.AddWithValue("@n", numericUpDown2.Value);
    cmd.Parameters.AddWithValue("@k", numericUpDown3.Value);
    cmd.Parameters.AddWithValue("@j", numericUpDown4.Value);
    cmd.Parameters.AddWithValue("@s", numericUpDown5.Value);
    cmd.Parameters.AddWithValue("@series1", series1Fordb());
    cmd.Parameters.AddWithValue("@number", vs.Count());
    cmd.Parameters.AddWithValue("@answer", series2Fordb());
    con.Open();
    cmd.ExecuteNonQuery();
    con.Close();
}
```

```
private void DeleteRecordFromMdb
    (string fileNameWithPath,string num)
{
    int number = Int32.Parse(num);
    var con = new OleDbConnection
        ("Provider = Microsoft.Jet.OLEDB.4.0; Data Source = "
            + fileNameWithPath);
    var cmd = new OleDbCommand();
    con.Open();
    cmd.Connection = con;
    cmd.CommandText = "DELETE FROM [my_table] " +
        "WHERE [order]=" + number + "";
    cmd.ExecuteNonQuery();
    con.Close();
}

private void DeleteAllRecordFromMdb
    (string fileNameWithPath)
{
    var con = new OleDbConnection
        ("Provider = Microsoft.Jet.OLEDB.4.0; Data Source = "
            + fileNameWithPath);
    var cmd = new OleDbCommand();
    con.Open();
    cmd.Connection = con;
    cmd.CommandText = "DELETE FROM [my_table] ";
    cmd.ExecuteNonQuery();
    con.Close();
}
```

3.2 Multi-Threading

We adopt multi-threading programming way. We split the program into two parts, which are the GUI part and the calculation part. In this way, even if the program haven't figured out, the window of the program won't be stick. The specific implemented function is bound in button2_Click.

```
private async void button2_Click(object sender, EventArgs e)
// Run buttom
{
    button2.Enabled = false;
    Algorithm algorithm = new Algorithm(
        (int)numericUpDown2.Value,
            (int)numericUpDown3.Value,
                (int)numericUpDown4.Value,
                    (int)numericUpDown5.Value,
                        totalList, judgeNumber);
    if (numericUpDown4.Value == numericUpDown5.Value)
    {
        vs= await Task.Run(()=>algorithm.ExecuteAlgorithm1());
    }
    else
    {
        vs = await Task.Run(()=>algorithm.ExecuteAlgorithm2());
    }
    //InsertToMdb(openFileDialog1.FileName);
    //UpdateToMdb(openFileDialog1.FileName);
    textBox3.Text = GetSeries2();
    //textBox3.Enabled = false;


}
```

4 Steps to Run the Program

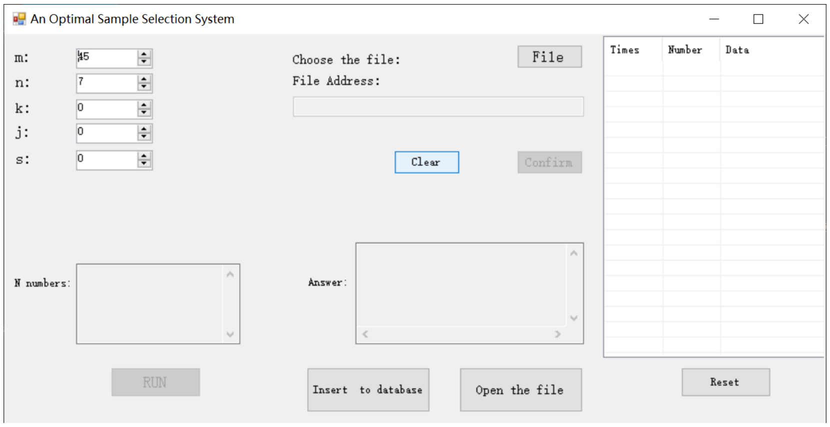
The detailed information of programming are listed in Table.4.0.1.

Attribute	Content
Operation System	**Windows SDK edition: 10.0**
Integrated Development Environment	**Visual Studio 2019(v142)**
Solution Settings	**Release in x86 Platform**
Optimization	**O2 optimize**
Programming Language	**C#**
Framework	**. Net framework 4.7.2**
Source Code hosting Platform	**Github**

Table. 4.0.1. Settings and Attribute

In order to make the operation more smooth, all the program environment and settings are completed and included in the file package. Just required to follow the steps below to run the program.

1. Open the package and find the *Information System Project.exe* file. Double-click the file to enter the program interface as Figure.4.0.1 exactly.

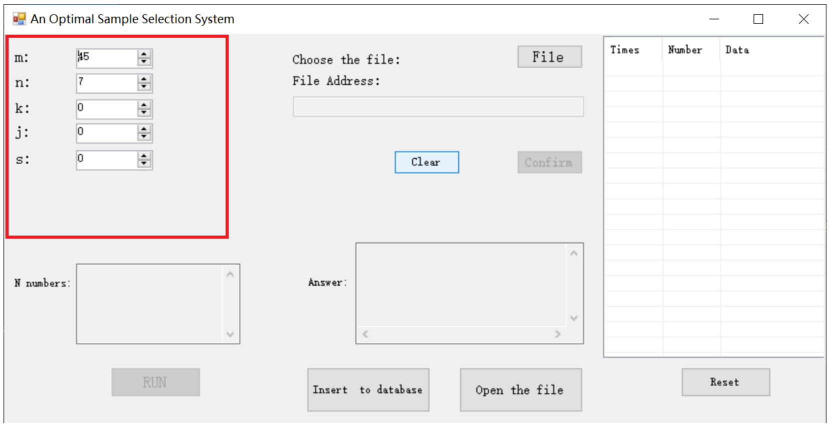


m: 45
n: 7
k: 0
j: 0
s: 0

Figure. 4.0.1. Initial GUI

In order to record the relevant output data of the program and facilitate display and modification later. It is required to create a *.mdb* file to store it, which is called *DataBase.mdb* in project package for example.

2. Choose the data of each parameter and input on the program surface as Figure.4.0.2.



m: 45
n: 7
k: 0
j: 0
s: 0

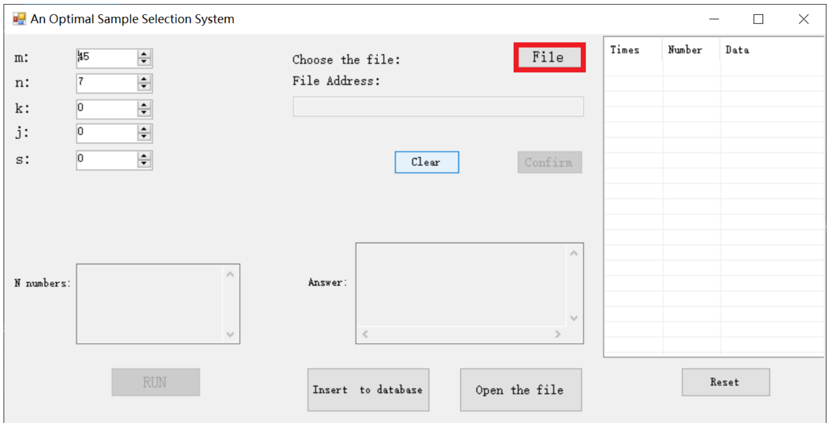
Figure. 4.0.2. Step1

3. Choose the DB file to store and operate the data, click the button **File** and choose the *.mdb* as Figure.4.0.3 In the previous step and **Confirm** if all get right.(**Clear** is a function that clear all the data you have input, including the parameter followed Figure.4.0.4)

4. Push the **RUN** button and the **N** number and final answer of your input will be shown on the surface window as Figure.4.0.5, you can check the answer after that.

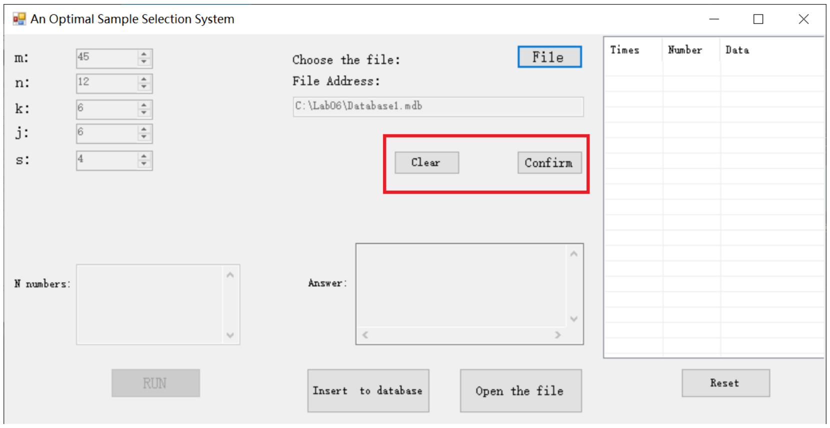
5. After confirming the data is correct, use the button **Insert to database** (Figure.4.0.6) to download the data on the DB file(*.mdb*), and the button **Open the file**(Figure.4.0.7) can open it to display the data you have calculated. It is also easy for you to delete or use any other operation on the data though your DB file.

6. After adding the data into your DB file, you can get the record of the message including order , calculated result and data information, which you can see the tips of first row. Moreover, you can choose the data which



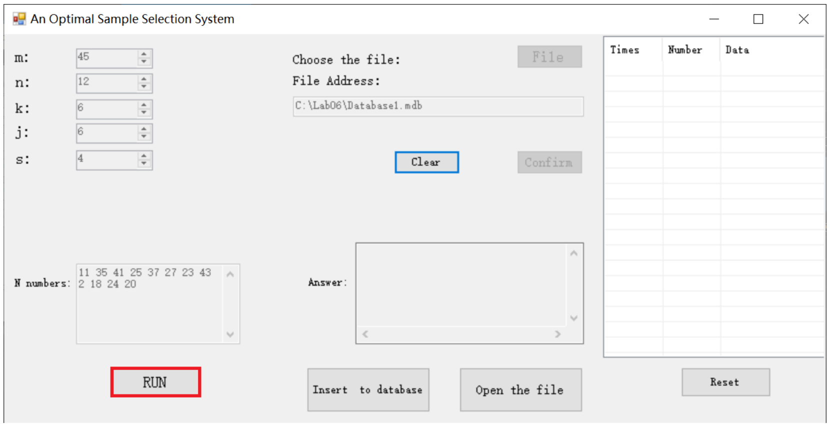
m: 45
n: 7
k: 0
j: 0
s: 0
File Address:

Figure. 4.0.3. Step2



m: 45
n: 12
k: 6
j: 6
s: 4
File Address: C:\Lab06\Database1.mdb

Figure. 4.0.4. Step3



m: 45
n: 12
k: 6
j: 6
s: 4

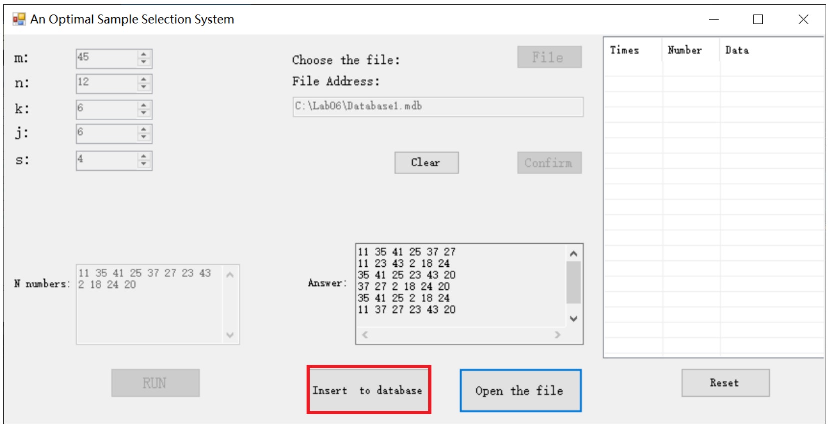
Choose the file:
File Address:
C:\Lab06\Database1.mdb

N numbers:
```
11 35 41 25 37 27 23 43
2 18 24 20
```

Answer:

Times	Number	Data

Figure. 4.0.5. Step4



m: 45
n: 12
k: 6
j: 6
s: 4

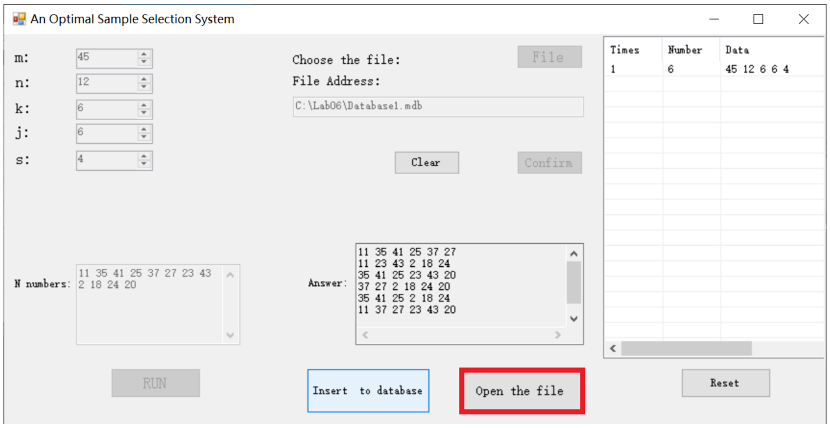
Choose the file:
File Address:
C:\Lab06\Database1.mdb

N numbers:
```
11 35 41 25 37 27 23 43
2 18 24 20
```

Answer:
```
11 35 41 25 37 27
11 23 43 2 18 24
35 41 25 23 43 20
37 27 2 18 24 20
35 41 25 2 18 24
11 37 27 23 43 20
```

Times	Number	Data

Figure. 4.0.6. Step5



m: 45
n: 12
k: 6
j: 6
s: 4

Choose the file:
File Address: C:\Lab06\Database1.mdb

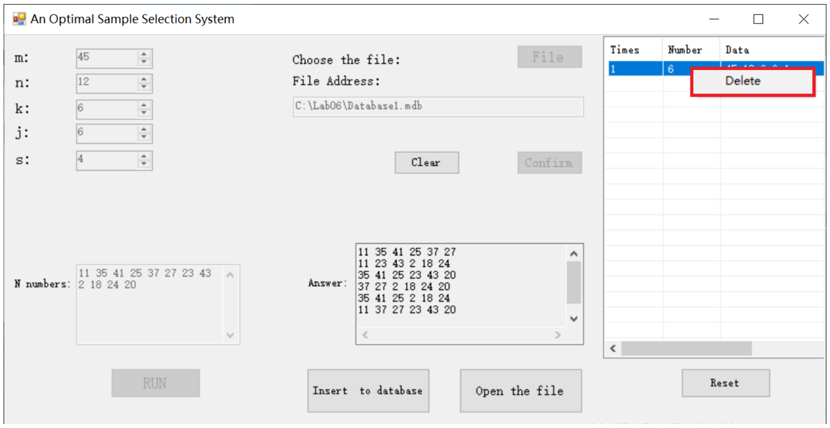
N numbers: 11 35 41 25 37 27 23 43 2 18 24 20

Answer:
```
11 35 41 25 37 27
11 23 43 2 18 24
35 41 25 23 43 20
37 27 2 18 24 20
35 41 25 2 18 24
11 37 27 23 43 20
```

Times	Number	Data
1	6	45 12 6 6 4

Figure. 4.0.7. Step6

you plan to delete by clicking the right button of mouse, then you can see the menu item **Delete** (Figure.4.0.8).



m: 45
n: 12
k: 6
j: 6
s: 4

Choose the file:
File Address: C:\Lab06\Database1.mdb

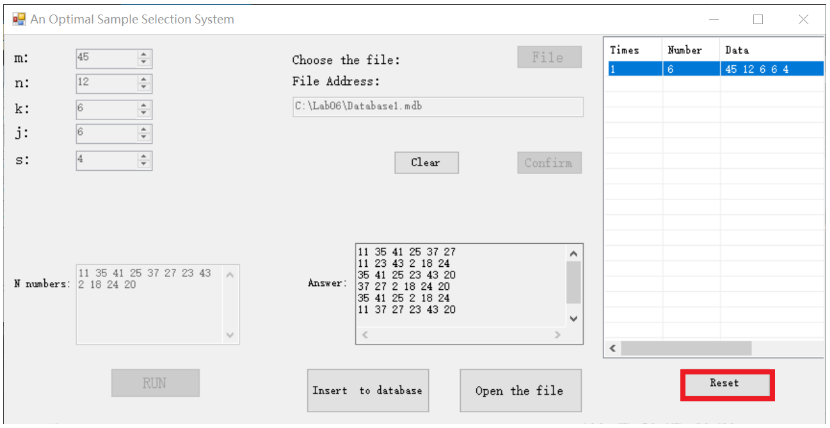
N numbers: 11 35 41 25 37 27 23 43 2 18 24 20

Answer:
```
11 35 41 25 37 27
11 23 43 2 18 24
35 41 25 23 43 20
37 27 2 18 24 20
35 41 25 2 18 24
11 37 27 23 43 20
```

Times	Number	Data
1	6	

Figure. 4.0.8. Step7

7. Besides, you can clear all the data from your DB file by using button **Reset** (Figure.4.0.9).



m: 45
n: 12
k: 6
j: 6
s: 4

Choose the file:
File Address:
C:\Lab06\Database1.mdb

N numbers: 11 35 41 25 37 27 23 43
2 18 24 20

Answer:

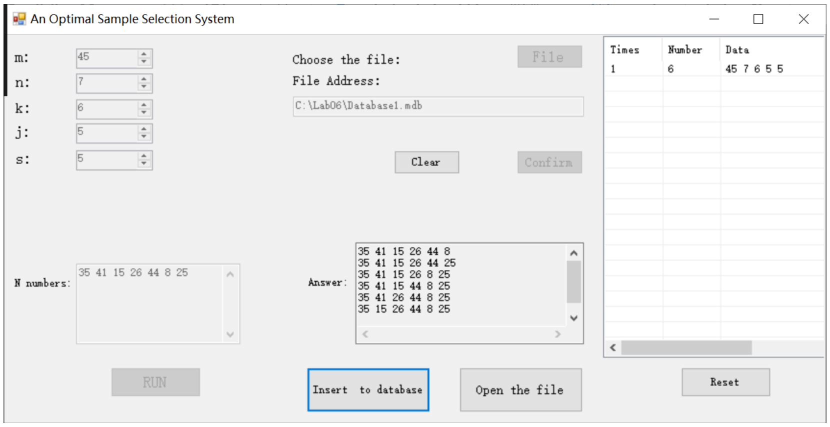
```
11 35 41 25 37 27
11 23 43 2 18 24
35 41 25 23 43 20
37 27 2 18 24 20
35 41 25 2 18 24
11 37 27 23 43 20
```

Times	Number	Data
1	6	45 12 6 6 4

Figure. 4.0.9. Step8

5 Program Test

If the program window can be displayed normally, you can enter the value for verification. The conditions of 1, 2, 3 and 4, 5 and 6, 7 in the project requirement file are similar, so we choose 1(Figure.5.0.1), 4(Figure.5.0.2), and 6(Figure.5.0.3) as the demo of our program.



m: 45
n: 7
k: 6
j: 5
s: 5

Choose the file:
File Address:
C:\Lab06\Database1.mdb

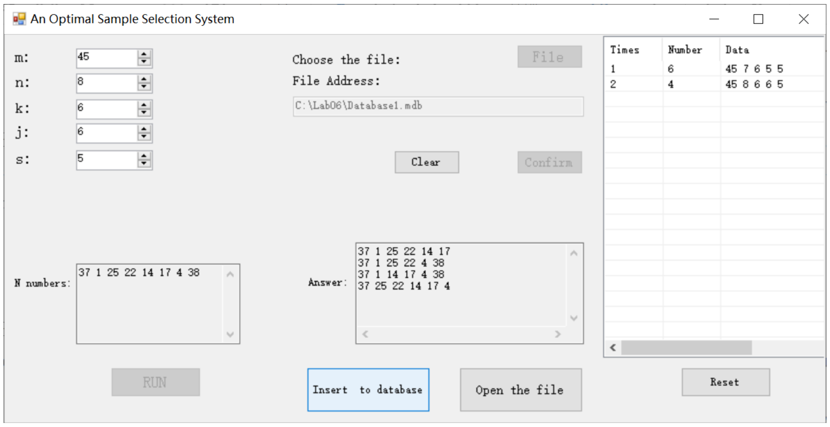
N numbers: 35 41 15 26 44 8 25

Answer:

```
35 41 15 26 44 8
35 41 15 26 44 25
35 41 15 26 8 25
35 41 15 44 8 25
35 41 26 44 8 25
35 15 26 44 8 25
```

Times	Number	Data
1	6	45 7 6 5 5

Figure. 5.0.1. E.g.1: Input the data: $m = 45, n = 7, k = 6, j = 5, s = 5$.



m: 45

n: 8

k: 6

j: 6

s: 5

Choose the file:

File Address: C:\Lab06\Database1.mdb

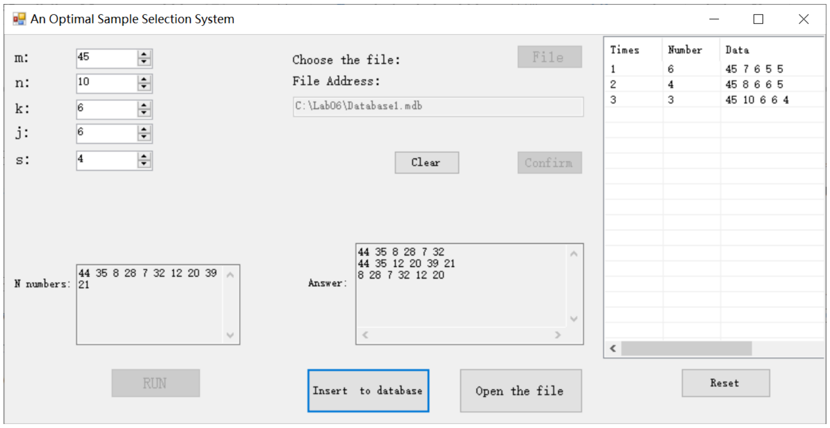
N numbers: 37 1 25 22 14 17 4 38

Answer:

```
37 1 25 22 14 17
37 1 25 22 4 38
37 1 14 17 4 38
37 25 22 14 17 4
```

Times	Number	Data
1	6	45 7 6 5 5
2	4	45 8 6 6 5

Figure. 5.0.2. E.g.4: Input the data: $m = 45, n = 8, k = 6, j = 6, s = 5$.



m: 45

n: 10

k: 6

j: 6

s: 4

Choose the file:

File Address: C:\Lab06\Database1.mdb

N numbers: 44 35 8 28 7 32 12 20 39 21

Answer:

```
44 35 8 28 7 32
44 35 12 20 39 21
8 28 7 32 12 20
```

Times	Number	Data
1	6	45 7 6 5 5
2	4	45 8 6 6 5
3	3	45 10 6 6 4

Figure. 5.0.3. E.g.6: Input the data: $m = 45, n = 10, k = 6, j = 6, s = 4$.

6 Summary

This project is based on the theoretical direction of **ME001** subject and combines some knowledge of data structure and mathematic, including optimal algorithms and combinatorics. But there are no correct understanding of some part of difficult and profound mathematic problems like fuzzy set. Authors point out about converse decimal digits to the binary make the big data abstraction in order to descend the time complexity. By using program, authors realize the process from theory to practice reflecting the theoretical view of unity of knowledge and practice. When writing large-scale projects, people often need cooperation and collaborative development, authors use **GitHub** for collaborative development and submit own patch code to collaborator's repository. In this article, we will utilize ideas to achieve team cooperation on **GitHub**. The last but not least, the goal of the future study and work is to work harder to learn this knowledge, in order to enrich, improve our level.

Appendices

A Programs for Algorithms

source/Algorithm.cs

```
using System;
using System.Collections.Generic;
using System.Collections.Specialized;
using System.ComponentModel;
using System.ComponentModel.Design;
using System.Diagnostics;
using System.Drawing.Drawing2D;
using System.Drawing.Text;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using System.Threading.Tasks;

namespace Information_System_Project
{
    public class Algorithm
    {
        private int n;
        private int k;
        private int j;
        private int s;
        private List<int> setNumberForK = new List<int>();
        private List<int> setNumberForJ = new List<int>();
        private Dictionary<int, int> dictionaryForAllSets
            = new Dictionary<int, int>();
        private Dictionary<int, int> dictionaryForAllSets2
            = new Dictionary<int, int>();
        private bool[] visit = new bool[10000000];
        private List<int> totalList = new List<int>();
        private Queue<int> queueForSet = new Queue<int>();
        private bool[] judgeNumber = new bool[46];
        private int[,] C = new int[26,26];
        public Algorithm(int n, int k, int j, int s,
            List<int> totalList, bool[] judgeNumber)
        {
            this.n = n;
            this.k = k;
            this.j = j;
            this.s = s;
            this.totalList = totalList;
            this.judgeNumber = judgeNumber;
        }
        public Queue<int> ExecuteAlgorithm1()
        {
            GreedyAlgorithm();
            return queueForSet;
        }
        public Queue<int> ExecuteAlgorithm2()
        {
            GreedyAlgorithm2();
            return queueForSet;
        }
        private void GreedyAlgorithm()
        {
            setNumberForK=CombinationForAllNum(n, k);
            int max;
            int now = 0;
            int allNum = TotalNumberForJ(n,j);
            int node;
            int index;
            List<int> vis = new List<int>();
            List<int> result = new List<int>();
            while (allNum > 0)
            {
                //Debug.WriteLine(allNum);
                max = 0;
                node = 0;
                index = 0;
                foreach (var element
                    in setNumberForK)
                {
                    int numOfUnfound = 0;
                    for (int j1 = (1 << s) - 1;
                        j1 <= element; j1++)
                    {
                        int cnt = 0;
                        for (int k1 = 0; k1 < n; k1++)
                        {
                            if ((j1 & (1 << k1)) != 0)
                            {
                                cnt++;
                            }
                        }
                        if ((j1 & element) == j1
                           && cnt == s &&
                                !dictionaryForAllSets.
                                    ContainsKey(j1))
                        {
                            numOfUnfound++;
                            vis.Add(j1);
                        }
                    }
                    if (max < numOfUnfound)
                    {
                        node = index;
                        max = numOfUnfound;
                        result.Clear();
                        foreach(var eachNum in vis)
                        {
                            result.Add(eachNum);
                        }
                    }
                    vis.Clear();
                    index++;
                }
                queueForSet.Enqueue(setNumberForK[node]);
                setNumberForK.RemoveAt(node);
                foreach(var eachNum in result)
```

```
                    {
                        dictionaryForAllSets[eachNum] = ++now;
                    }
                    allNum -= max;

                }

        }
        private void GreedyAlgorithm2()
        {
            setNumberForK=CombinationForAllNum(n, k);
            int max;
            int now = 0;
            int allNum = TotalNumberForJ(n, j);
            int node;
            int index;
            List<int> vis = new List<int>();
            List<int> result = new List<int>();
            while (allNum > 0)
            {
                max = 0;
                node = 0;
                index = 0;
                foreach (var element in setNumberForK)
                {
                    int numOfUnfound = 0;
                    for (int j1 = (1 << s) - 1;
                        j1 <= element; j1++)
                    {
                        int cnt = 0;
                        var answer = 0;
                        for (int k1 = 0; k1 < n; k1++)
                        {
                            if ((j1 & (1 << k1)) != 0)
                            {
                                cnt++;
                            }
                        }
                        if ((j1 & element) == j1
                            && cnt == s &&
                                !dictionaryForAllSets.
                                    ContainsKey(j1))
                        {
                            int num = j - s;
                            numOfSetContainj1
                                (0,j1, num,ref answer);
                            numOfUnfound += answer;
                            vis.Add(j1);
                        }
                    }
                    foreach(var eachNum in setNumberForJ)
                    {
                        dictionaryForAllSets2.Remove(eachNum);
                    }
                    setNumberForJ.Clear();
                    if (max < numOfUnfound)
                    {
                        max = numOfUnfound;
                        node = index;
                        result.Clear();
                        foreach(var eachNum in vis)
                        {
                            result.Add(eachNum);
                        }
                    }
                    index++;
                    vis.Clear();
                }
                //Debug.WriteLine(max);
                SetDictionary2(result);
                queueForSet.Enqueue(setNumberForK[node]);
                foreach(var eachNum in result)
                {
                    //Debug.Write(eachNum + " ");
                    dictionaryForAllSets[eachNum] = ++now;
                }
```

```
                setNumberForK.RemoveAt(node);
                allNum -= max;
            }
        }
        private void numOfSetContainj1(int node, int j1,
            int num,ref int answer)
        {
            if (num == 0 &&
                !dictionaryForAllSets2.
                    ContainsKey(j1))
            {
                answer += 1;
                setNumberForJ.Add(j1);
                dictionaryForAllSets2[j1] =
                    dictionaryForAllSets2.Count() + 1;
                return;
            }
            else if (num == 0 &&
                dictionaryForAllSets2.
                    ContainsKey(j1))
                return;
            for(int i1 = node; i1 < n; i1++)
            {
                if(((1<<i1) & j1) == 0)
                {
                    numOfSetContainj1(i1+1, j1 | (1 << i1),
                        num - 1, ref answer);
                }
            }
        }

        private void SetDictionary2(List<int> result)
        {
            foreach(var eachNum in result)
            {
                FindEachElement(0,eachNum,j-s);
            }
        }

        private void FindEachElement(int node,int element,
            int num)
        {
            if (num == 0 &&
                !dictionaryForAllSets2.
                    ContainsKey(element))
            {
                dictionaryForAllSets2[element]
                    = dictionaryForAllSets2.Count() + 1;
                return;
            }
            else if (num == 0 &&
                dictionaryForAllSets2.
                    ContainsKey(element))
                return;
            for (int i1 = node; i1 < n; i1++)
            {
                if (((1 << i1) & element) == 0)
                {
                    FindEachElement(i1, element | (1 << i1),
                        num - 1);
                }
            }
        }

        private List<int> CombinationForAllNum(int n,int k)
        {
            List<int> Combination = new List<int>();
            for (int i = (1<<(k))-1; i < (1 << n); i++)
            {
                var cnt = 0;
                for (int j1 = 0; j1 < n; j1++)
                {
                    if ((i & (1 << j1)) != 0)
                    {
                        cnt++;
                    }
                }
                if (cnt == k)
                {
                    Combination.Add(i);
                    //myBV.Add(new BitVector32(i));
                }
            }
            return Combination;
        }
        private int TotalNumberForJ(int n,int j)
        {
            for(int i = 0; i <= n; i++)
            {
                for(int m = 0; m <= j; m++)
                {
                    C[i, m] = 0;
                }
            }
            for(int i = 0; i <= n; i++)
            {
                C[i, 0] = 1;
                if (i == n && j == 0)
                    return C[n, j];
            }
            for (int i = 1; i <= n; i++)
            {
                for (int m = 1; m <= i; m++)
                {
                    C[i, m] = C[i - 1, m - 1] + C[i-1, m];
                    if (i == n && m == j)
                        return C[n, m];
                }
            }
            return C[n, j];
        }

        private void Dfs(int start,int setNum,
            int currentNumber,int totalNum,ref int result)
        {

            int now = currentNumber;
            List<int> vis = new List<int>();
            if (setNum >= result)
                return;
            if (currentNumber == totalNum)
            {

                //if (setNum < result)
                result = setNum;
                return;
            }
            for (int i = start; i < setNumberForK.Count; i++)
            {
                if (!visit[i])
                {
                    for (int j1 = (1 << s) - 1;
                        j1 <= setNumberForK[i]; j1++)
                    {
                        int cnt = 0;
                        for (int k1 = 0; k1 < n; k1++)
                        {
                            if ((j1 & (1 << k1)) != 0)
                            {
                                cnt++;
                            }
                        }
                        if ((j1 & setNumberForK[i])==j1
                            && cnt == s &&
                                !dictionaryForAllSets.
                                    ContainsKey(j1))
                        {
                            now++;
                            dictionaryForAllSets[j1] = now;
                            vis.Add(j1);
                        }
                    }
                    //Debug.WriteLine(" ");
                    Dfs(i + 1, setNum + 1, now,
                        totalNum, ref result);
                    visit[i] = false;
                    now = currentNumber;

                }
                foreach (var eachNum in vis)
                {
                    dictionaryForAllSets.Remove(eachNum);
                }
                vis.Clear();
            }
            return ;
        }
    }
}
```

B Programs for GUI

source/Form1.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.OleDb;
using System.Diagnostics;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Information_System_Project
{
    public partial class Form1 : Form
    {
        int cnt = 1;
        Queue<int> vs;
        List<int> totalList = new List<int>();
        bool[] judgeNumber = new bool[55];
        OpenFileDialog openFileDialog1
            = new OpenFileDialog();

        public Form1()
        {
            InitializeComponent();
            listView1.View = View.Details;
            listView1.GridLines = true;
            listView1.FullRowSelect = true;
            listView1.Columns.Add(”Times”,60);
            listView1.Columns.Add(”Number”, 60);
            listView1.Columns.Add(”Data”,244);
        }


        private void numericUpDown3_ValueChanged
            (object sender, EventArgs e)
        {
            numericUpDown3.Maximum
                = numericUpDown2.Value;

        }

        private void numericUpDown4_ValueChanged
            (object sender, EventArgs e)
        {
            numericUpDown4.Maximum
                = numericUpDown3.Value;
        }

        private void numericUpDown5_ValueChanged
            (object sender, EventArgs e)
        {
            numericUpDown5.Maximum
                = numericUpDown4.Value;
        }

        private void button1_Click
            (object sender, EventArgs e)//Confim button
        {
            button1.Enabled=false;
            ChooseTotalList();
            //DisableNumericUpDown();
            textBox2.Enabled = false;
            button1.Enabled = false;
            button3.Enabled = false;
            button2.Enabled = true;
        }

        private async void button2_Click
            (object sender, EventArgs e)// Run buttom
        {
            button2.Enabled = false;
            Algorithm algorithm =
                new Algorithm(
                    (int)numericUpDown2.Value,
                    (int)numericUpDown3.Value,
                    (int)numericUpDown4.Value,
                    (int)numericUpDown5.Value,
                    totalList, judgeNumber);
            if (numericUpDown4.Value
                == numericUpDown5.Value)
            {
                vs= await Task.Run
                    (()=>algorithm.ExecuteAlgorithm1());
            }
            else
            {
                vs = await Task.Run
                    (()=>algorithm.ExecuteAlgorithm2());
            }
            //InsertToMdb(openFileDialog1.FileName);
            //UpdateToMdb(openFileDialog1.FileName);
            textBox3.Text = GetSeries2();
            //textBox3.Enabled = false;


        }

        private void button3_Click
            (object sender, EventArgs e)// File button
        {

            openFileDialog1.InitialDirectory = "c:\\";
            openFileDialog1.Filter
                = "Database files (*.mdb)|*.mdb";
            openFileDialog1.FilterIndex = 0;
            openFileDialog1.RestoreDirectory = true;


            if (openFileDialog1.ShowDialog()
                == DialogResult.OK)
            {
                DisableNumericUpDown();
                textBox1.Enabled = false;
                button1.Enabled = true;
                textBox2.Text = openFileDialog1.FileName;
                CreateTableInToMdb
                    (openFileDialog1.FileName);
            }

        }

        private void button4_Click
            (object sender, EventArgs e)//Clear function
        {
            InitializeFunctionForClear();
        }

        private void button5_Click
            (object sender, EventArgs e)
            //Open the file button
        {
            Process proc = new Process();
            proc.EnableRaisingEvents = false;
            proc.StartInfo.FileName = openFileDialog1.FileName;
            proc.Start();
        }

        private void button6_Click
            (object sender, EventArgs e)
            //Insert to database button
        {
            InsertToMdb(openFileDialog1.FileName);
            string[] arr = new string[3];
            arr[0] = cnt.ToString();
            arr[1] = vs.Count.ToString();
            arr[2] = numericUpDown1.Value.ToString()
                 + " "
                + numericUpDown2.Value.ToString() +
                " " + numericUpDown3.Value.ToString()
                + " "
                + numericUpDown4.Value.ToString()
                + " "
                + numericUpDown5.Value.ToString();
            ListViewItem itm=new ListViewItem(arr);
            listView1.Items.Add(itm);
            cnt++;
        }

        private void button7_Click
            (object sender, EventArgs e)//reset button
        {
            listView1.Items.Clear();
            cnt = 1;
            DeleteAllRecordFromMdb
                (openFileDialog1.FileName);
        }

        private void ChooseTotalList()
        {
            InitializeJudgeNumber();
            totalList.Clear();
            var rand = new Random();
            StringBuilder str = new StringBuilder();
            for (int i = 1;
                 i <= numericUpDown2.Value; i++)
            {
                int randNumber = rand.Next
                    (1, (int)numericUpDown1.Value + 1);
                if (!judgeNumber[randNumber])
                {
                    judgeNumber[randNumber] = true;
                    totalList.Add(randNumber);
                    str.Append
                        (totalList[i - 1].ToString());
                    str.Append(" ");
                }
                else
                {
                    i--;
                }
            }
            textBox1.Text = str.ToString();
        }
        private void InitializeJudgeNumber()
        {
            for (int i = 0; i < judgeNumber.Length; i++)
                judgeNumber[i] = false;
        }

        private void InitializeFunctionForClear()
        {
            EnableNumericUpDown();
            button1.Enabled = true;
            textBox1.Enabled = true;
            textBox1.Clear();
            //textBox2.Enabled = true;
            //textBox2.Clear();
            textBox3.Clear();
            button3.Enabled = true;
        }

        private void DisableNumericUpDown()
        {
            numericUpDown1.Enabled = false;
            numericUpDown2.Enabled = false;
            numericUpDown3.Enabled = false;
            numericUpDown4.Enabled = false;
            numericUpDown5.Enabled = false;
        }

        private void EnableNumericUpDown()
        {
            numericUpDown1.Enabled = true;
            numericUpDown2.Enabled = true;
            numericUpDown3.Enabled = true;
            numericUpDown4.Enabled = true;
            numericUpDown5.Enabled = true;
        }

        private void CreateTableInToMdb
            (string fileNameWithPath)
        {
            try
            {
                OleDbConnection myConnection =
                    new OleDbConnection
                        ("Provider=Microsoft.Jet.OLEDB.4.0; Data Source=
                        fileNameWithPath);
                myConnection.Open();
                OleDbCommand myCommand
                    = new OleDbCommand();
                myCommand.Connection = myConnection;
                myCommand.CommandText
                    = "CREATE TABLE my_table" +
                    "([order] NUMBER, "+
                    "[m] NUMBER, " +
                    "[n] NUMBER, " +
                    "[k] NUMBER, " +
                    "[j] Number," +
                    "[s] NUMBER, " +
                    "[n numbers] TEXT," +
                    "[minium number of sets] NUMBER, " +
                    "[answer] TEXT)";
                myCommand.ExecuteNonQuery();
                myCommand.Connection.Close();
            }
            catch { }
        }

        private void InsertToMdb(string fileNameWithPath)
        {
            var con = new OleDbConnection
                ("Provider = Microsoft.Jet.OLEDB.4.0; Data Source = "
                + fileNameWithPath);
            var cmd = new OleDbCommand();
            cmd.Connection = con;
            cmd.CommandText = "insert into my_table "+
                "([order],[m],[n],[k],[j],[s],"+
                "[n numbers],[minium number of sets], [answer]) "
                +"values (@order, @m, @n, "+
                "@k,@j,@s,@series1 , @number, @answer);";
            cmd.Parameters.AddWithValue("@order", cnt);
            cmd.Parameters.AddWithValue
                ("@m", numericUpDown1.Value);
            cmd.Parameters.AddWithValue
                ("@n", numericUpDown2.Value);
            cmd.Parameters.AddWithValue
                ("@k", numericUpDown3.Value);
            cmd.Parameters.AddWithValue
```

```
                ("@j", numericUpDown4.Value);
            cmd.Parameters.AddWithValue
                ("@s", numericUpDown5.Value);
            cmd.Parameters.AddWithValue
                ("@series1", series1Fordb());
            cmd.Parameters.AddWithValue
                ("@number", vs.Count());
            cmd.Parameters.AddWithValue
                ("@answer", series2Fordb());
            con.Open();
            cmd.ExecuteNonQuery();
            con.Close();
        }

        private void DeleteRecordFromMdb
            (string fileNameWithPath,string num)
        {
            int number = Int32.Parse(num);
            var con = new OleDbConnection
                ("Provider = Microsoft.Jet.OLEDB.4.0; "+
                "Data Source = " + fileNameWithPath);
            var cmd = new OleDbCommand();
            con.Open();
            cmd.Connection = con;
            cmd.CommandText = "DELETE FROM [my_table] " +
                "WHERE [order]=" + number + "";
            cmd.ExecuteNonQuery();
            con.Close();
        }

        private void DeleteAllRecordFromMdb
            (string fileNameWithPath)
        {
             var con = new OleDbConnection
                ("Provider = Microsoft.Jet.OLEDB.4.0; "+
                "Data Source = " + fileNameWithPath);
            var cmd = new OleDbCommand();
            con.Open();
            cmd.Connection = con;
            cmd.CommandText = "DELETE FROM [my_table] ";
            cmd.ExecuteNonQuery();
            con.Close();
        }

        private string series1Fordb()
        {
            string series1 = "";
            foreach (var num in totalList)
            {
                series1 += num.ToString();
                series1 += " ";
            }
            return series1;
        }

        private string series2Fordb()
        {
            string series2 = "";
            int index = 0;
            foreach(var num in vs)
            {
                if (index != 0)
                    series2 += ";  ";
                for (int i = 0;
                    i < numericUpDown2.Value; i++)
                {
                    if (((1 << i) & num) != 0)
                    {
                        series2 +=
                            totalList[i].ToString();
                        series2 += " ";
                    }
                }

                index++;
            }
            return series2;
        }

        private string GetSeries2()
        {
            string series2 = "";
            foreach(var num in vs)
            {
                for(int i = 0;
                    i < numericUpDown2.Value; i++)
                {
                    if (((1 << i) & num)!=0)
                    {
                        series2 +=
                            totalList[i].ToString();
                        series2 += " ";
                    }
                }
                series2 += "\r\n";
            }
            return series2;
        }

        private void listView1_SelectedIndexChanged
            (object sender, EventArgs e)
        {
            //var selectedItemText = "+
            //"(listView1.SelectedItem ?? "(none)")
            //.ToString();
            //MessageBox.Show"
            //+"("Selected: " + selectedItemText);

        }

        private void listView1_MouseDown
            (object sender, MouseEventArgs e)
        {
            if (listView1.SelectedItems.Count >= 1
                && e.Button==MouseButtons.Right)
            {
                ListViewItem item = listView1.SelectedItems[0];

    //here i check for the Mouse pointer location on click if its conta
    // in the actual selected item's bounds or not .
    // cuz i ran into a problem with the ui once because of that ..
                if (item.Bounds.Contains(e.Location))
                {
                    ContextMenu cm = new ContextMenu();
                    MenuItem menuItemForDelete
                        = new MenuItem();
                    menuItemForDelete.Text = "Delete";
                    menuItemForDelete.Click +=
                         new EventHandler
                            (menuItemForDelete_Click);
                    cm.MenuItems.Add(menuItemForDelete);
                    listView1.ContextMenu = cm;
                }
            }
        }

        private void menuItemForDelete_Click
            (object sender,EventArgs e)
        {
            var element = listView1.SelectedItems[0];
            DeleteRecordFromMdb
                (openFileDialog1.FileName,
                    element.SubItems[0].Text);
            listView1.Items.Remove
                (listView1.SelectedItems[0]);

        }
    }

}
```