

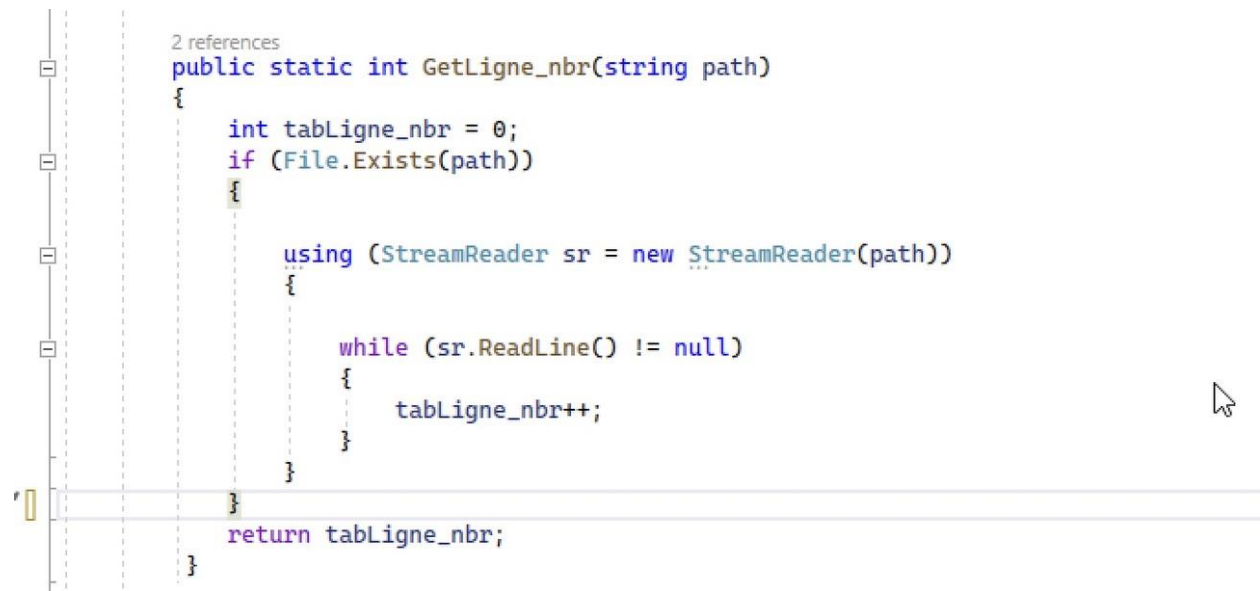
## Compte Rendu

Amal Khoubaba

AYMANE HINANE

4 Année G5 Emsi centre

Cette fonction elle sert a obtenir le nombre de ligne de ma matrice



```
2 references
public static int GetLigne_nbr(string path)
{
    int tabLigne_nbr = 0;
    if (File.Exists(path))
    {
        using (StreamReader sr = new StreamReader(path))
        {
            while (sr.ReadLine() != null)
            {
                tabLigne_nbr++;
            }
        }
    }
    return tabLigne_nbr;
}
```

Cette fonction elle sert à obtenir le nombre de colonne de ma matrice

```

public static int GetColumn_nbr(string path)
{
    int tabColumn_nbr = 0;
    string? txt = null;

    if (File.Exists(path))
    {
        using (StreamReader sr = new StreamReader(path))
        {
            while ((txt=sr!.ReadLine()) != null)
            {
                foreach (var c in txt)
                {
                    if (c != '-')
                        tabColumn_nbr++;
                }
                break;
            }
        }

        return tabColumn_nbr;
    }
}

```

Cette fonction lie une matrice d' un fichier est la stock dans une  
matrice[GetLigne\_nbr][GetColumn\_nbr]

```

3 references
public static int[][] ReadMatrix(string path)
{
    int nbr_ligne = GetLigne_nbr(path);
    int nbr_column = GetColumn_nbr(path);
    string? txt = null;
    int[][] Matrix = new int[nbr_ligne][];
    int index = 0;

    using (StreamReader sr = new StreamReader(path))
    {
        while((txt=sr.ReadLine())!=null)
        {
            Matrix[index] = new int[nbr_column];
            for(int i=0,j=0;i<txt.Length;i++)
            {
                if(i>=1)
                {
                    if (txt[i - 1] == '-')
                        continue;

                    if (txt[i] != '-')
                    {
                        Matrix[index][j] = int.Parse(txt[i].ToString());
                        j++;
                    }
                    else
                    {
                        Matrix[index][j] = int.Parse(txt[i].ToString()+ txt[i+1].ToString());
                        j++;
                    }
                }
            }
            index++;
        }
    }
    return Matrix;
}

```

Je compare le mot avec la matrice on fonction du langage

```

1 reference
private static bool Alphabet_Validation(string mots)
{
    Func<char, int> GetIndex = (char c) => c == 'a' ? 0 : c == 'b' ? 1 : 2;
    int[][] Matrix = ReadMatrix(path);

    for(int i=0;i<mots.Length;i++)
    {
        if (Matrix[i][GetIndex(mots[i])] == -1)
            return false;
    }

    //return Matrix[mots.Length - 1][GetIndex(mots[mots.Length - 1])] == mots.Length ? true:false;
    return true;
}

```

```

1 reference
private static bool Numeric_validation(string mots)
{
    int[][] Matrix = ReadMatrix(path);

    for (int i = 0; i < mots.Length; i++)
    {
        if (Matrix[i][int.Parse(mots[i].ToString())] == -1 )
            return false;
    }

    return true;
}

```

Si je tombe sur -1 ça veut dire que mon mots n'appartient pas au langage

Je dois vérifier le vocabulaire de ma matrice si c'est un alphabet ou entier

```
1 reference
private static string Vocabulaire(string mots)
{
    bool isAlphabet=true,isNumeric=true;

    byte[] asciiBytes = Encoding.ASCII.GetBytes(mots);

    foreach (var ascii in asciiBytes)
    {
        if (ascii < 97 || ascii > 99)
        {
            isAlphabet = false;
            break;
        }
    }
    if (isAlphabet)
        return "alphabet";
    else
    {
        foreach (var ascii in asciiBytes)
        {
            if (ascii < 48 || ascii > 57)
            {
                isNumeric = false;
                break;
            }
        }

        if (isNumeric)
            return "numeric";
        return "none";
    }
}
```

Function Principal

1 reference

```

private void compile_Click(object sender, EventArgs e)
{
    string path = @"C:\Users\aymane\Desktop\WinFormsApp1\WinFormsApp1\File1.txt";
    int[][] Matrix = ReadMatrix(path);

    var mots = this.mots.Text;

    if(mots.Length > GetLigne_nbr(path))
    {
        this.message.Text = "mots trop long";
        return;
    }

    string vocabulaire = Vocabulaire(mots);

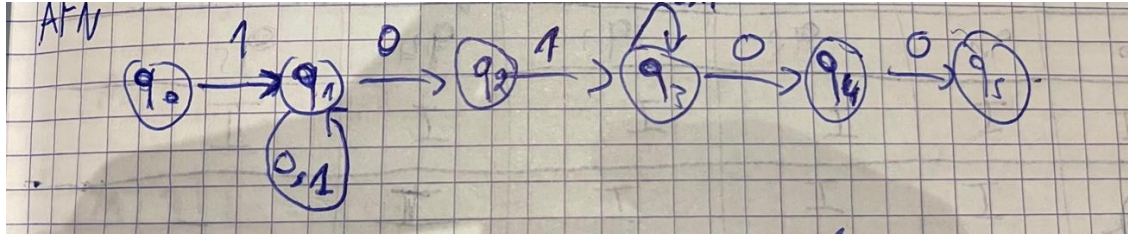
    if(vocabulaire == "numeric")
        this.message.Text = Numeric_validation(mots) ? "mots valide" : "mots non valide";
    else if(vocabulaire == "alphabet")
        this.message.Text = Alphabet_Validation(mots) ? "mots valide" : "mots non valide";
}

```

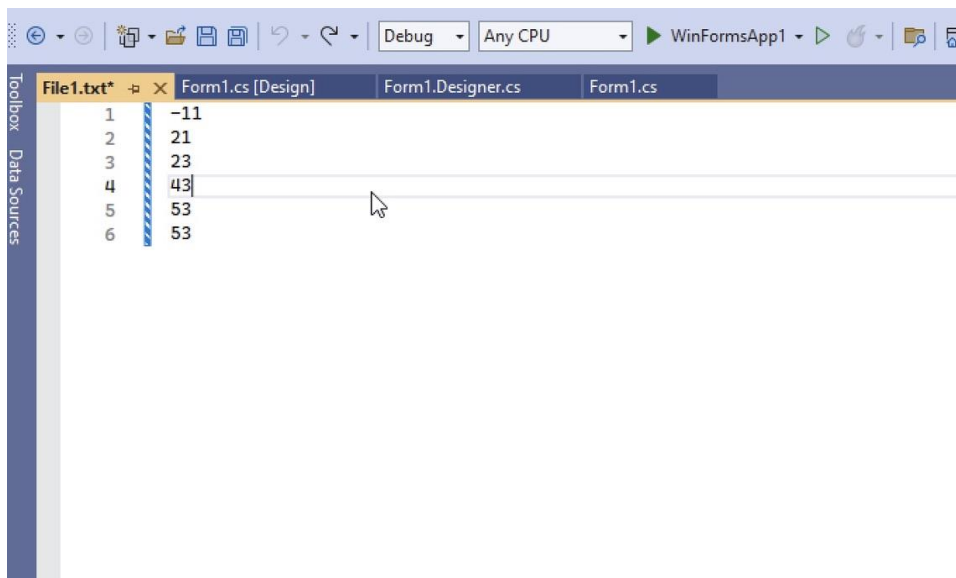
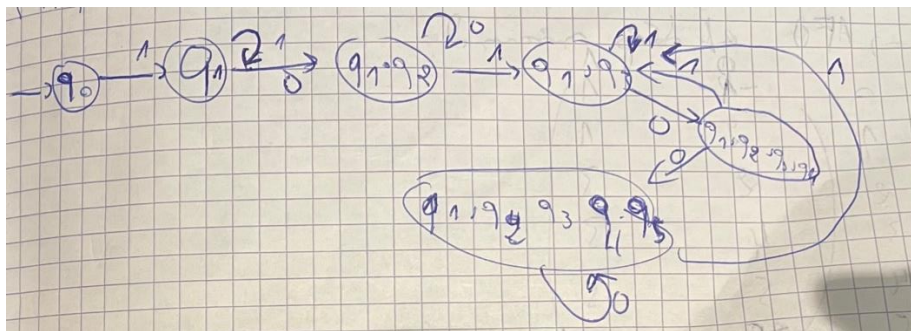
Ex 1:

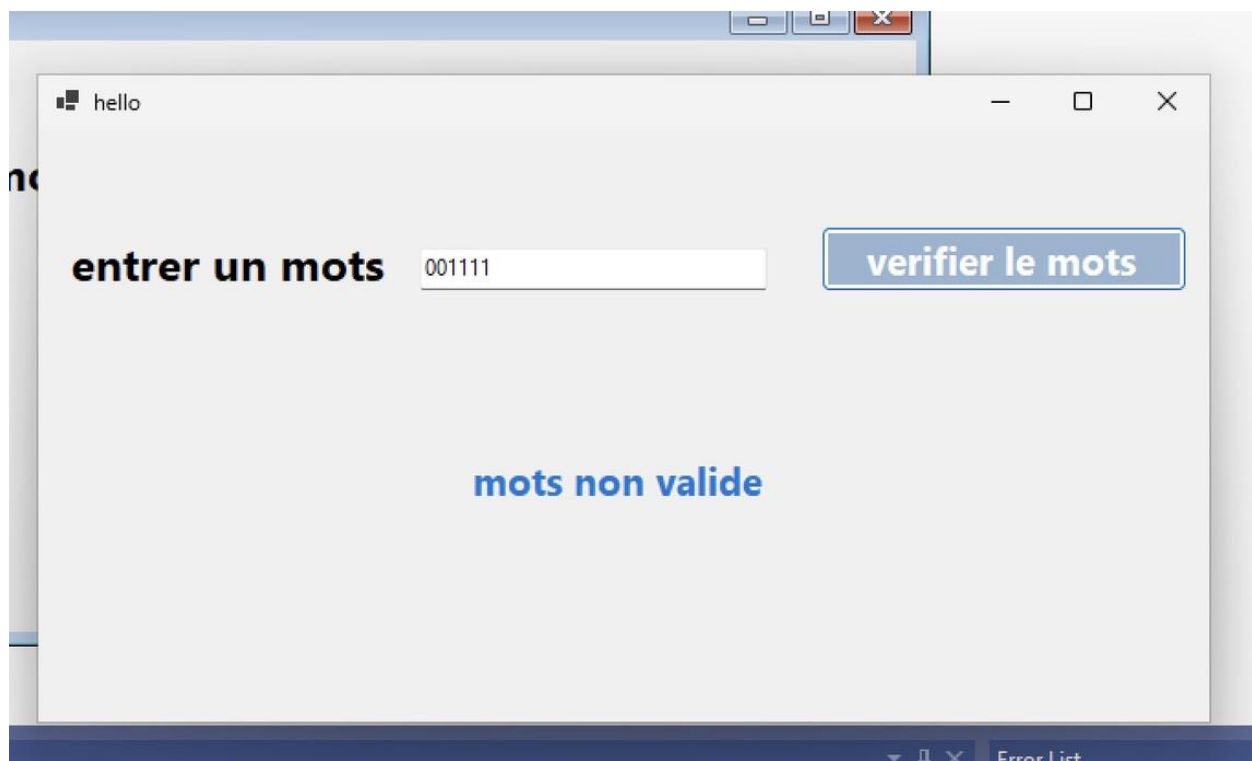
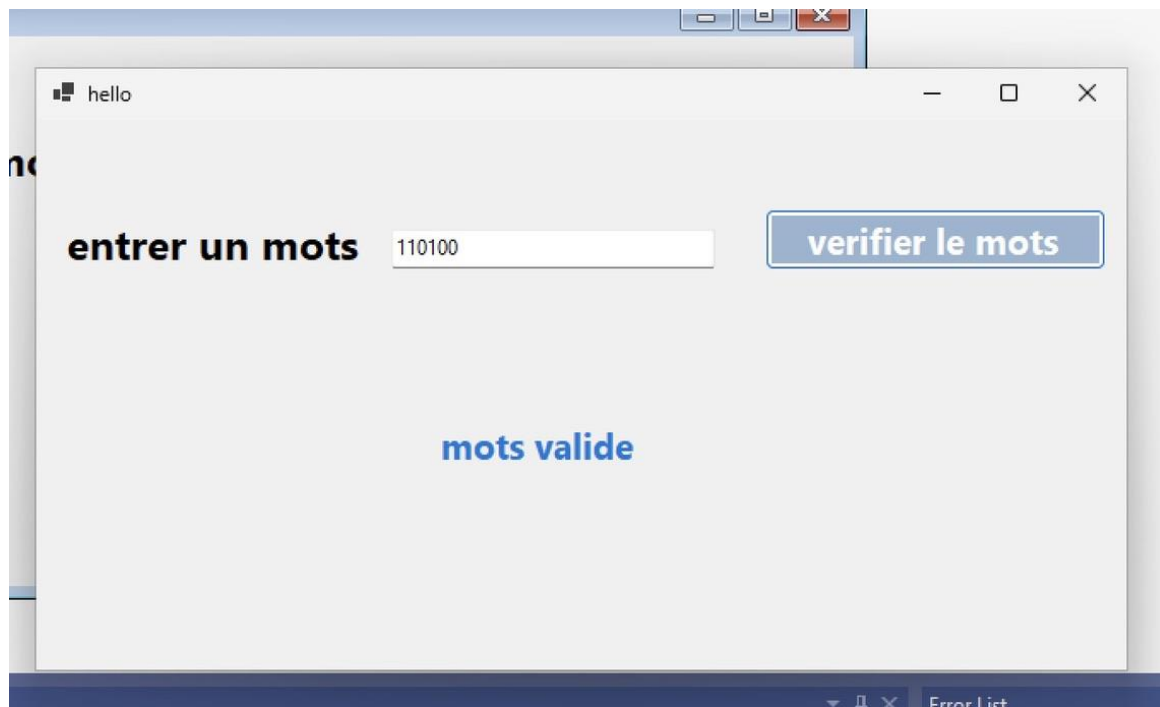
1.(0+1)\*01(0+1)\*00

AFD



AFDM



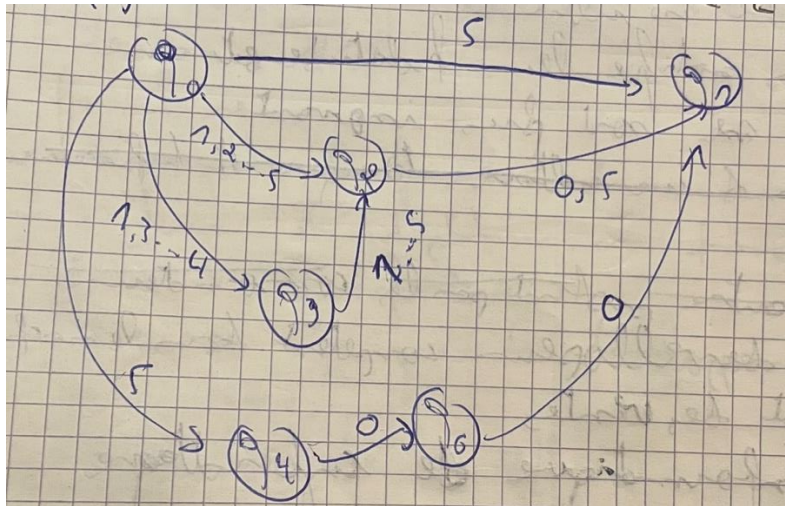


Ex2:

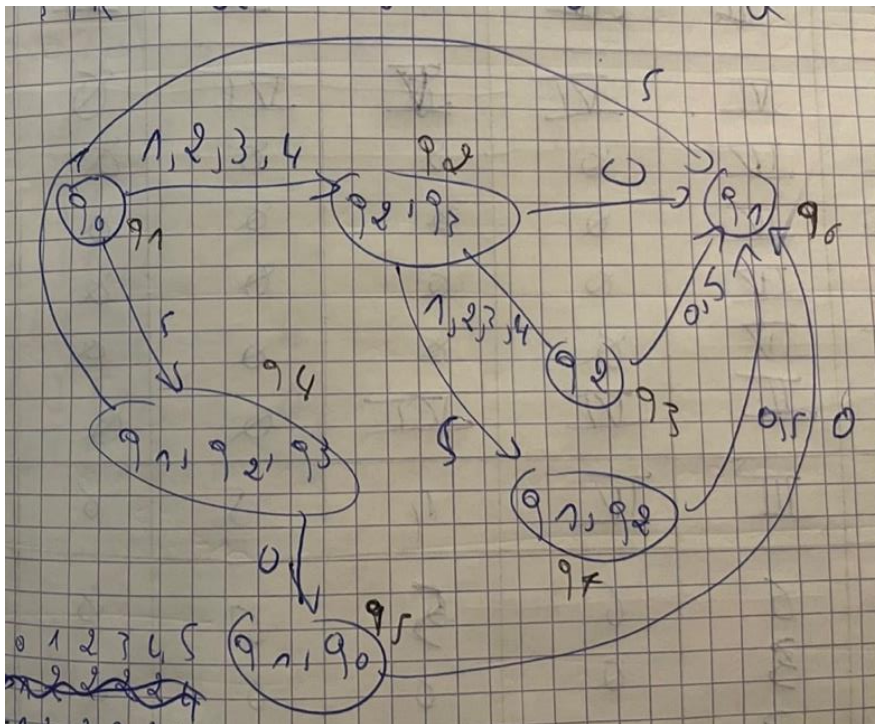


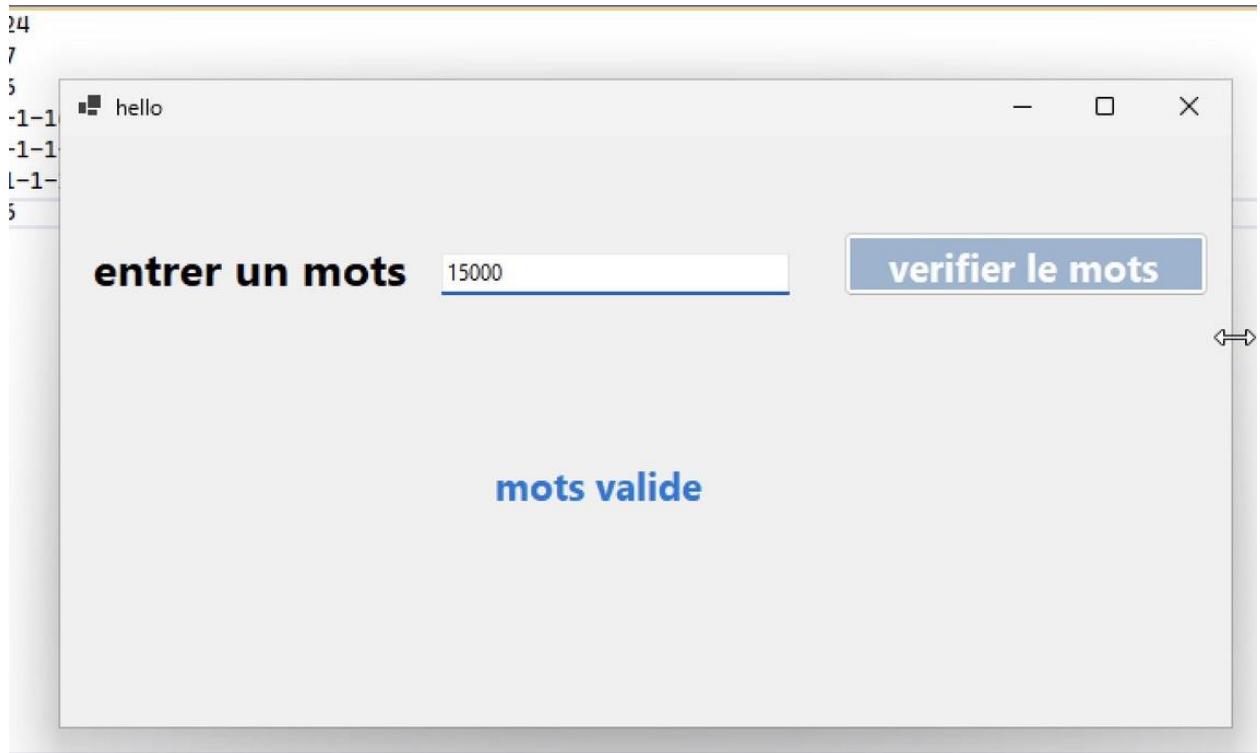
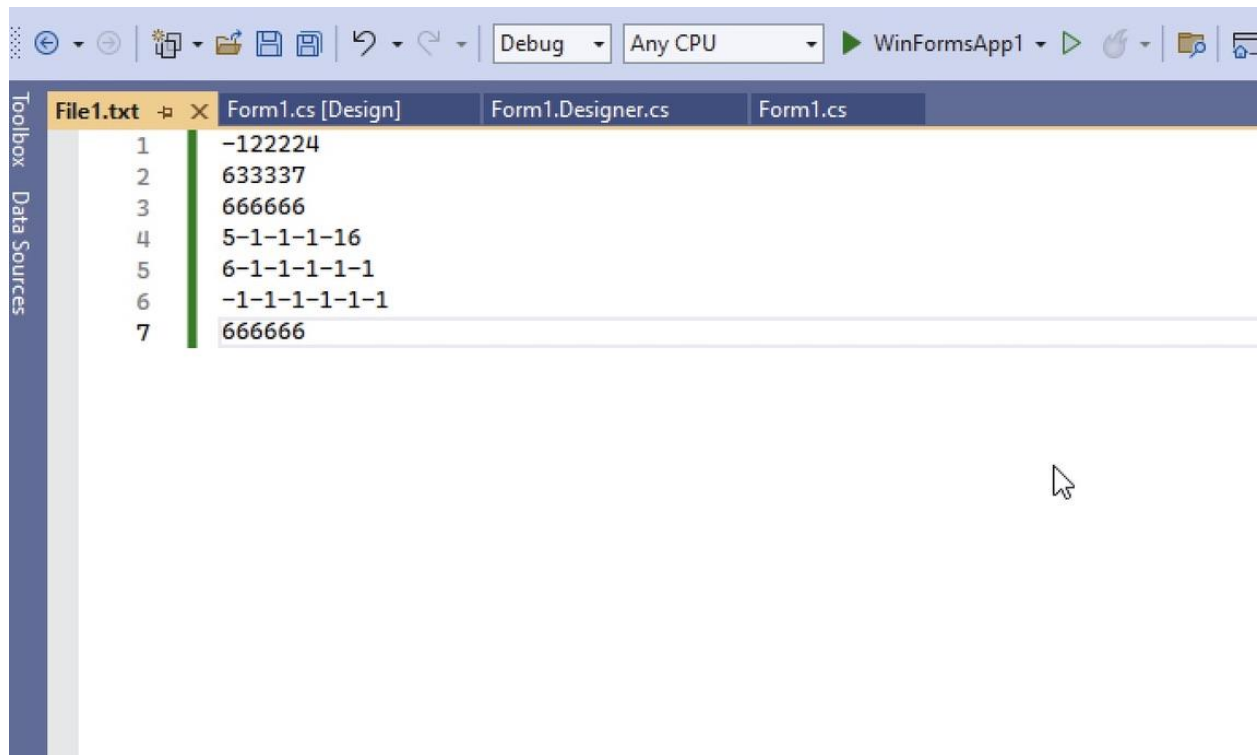
$$5 + [1-5](0+5) + [1-4][1-5](0+5) + 500$$

AFD



AFDM





24

7

5

-1-1

-1-1

-1-1

5

hello

entrer un mots

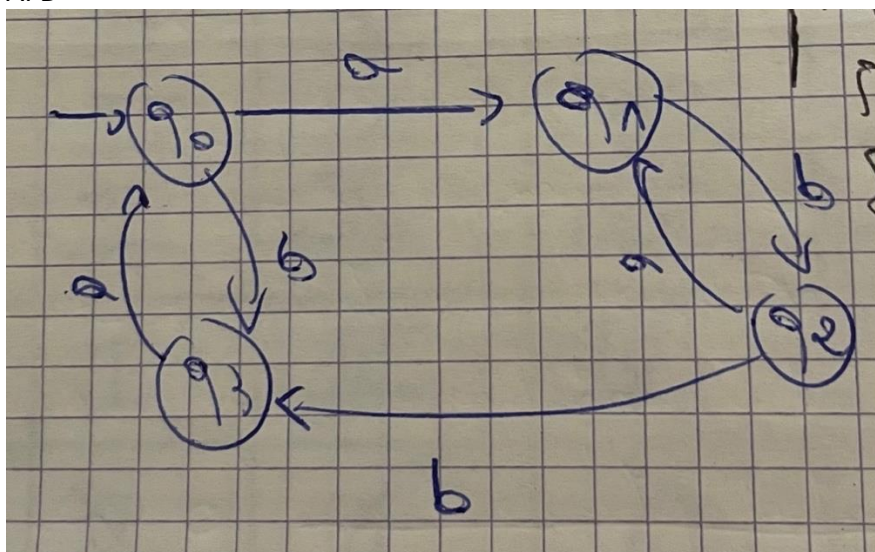
verifier le mots

mots non valide

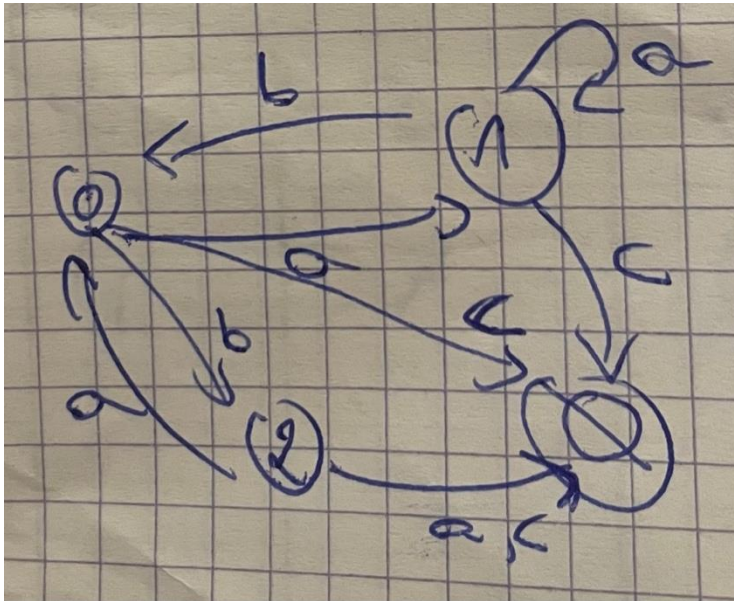
Ex3

$(a+ab+ba)^*$

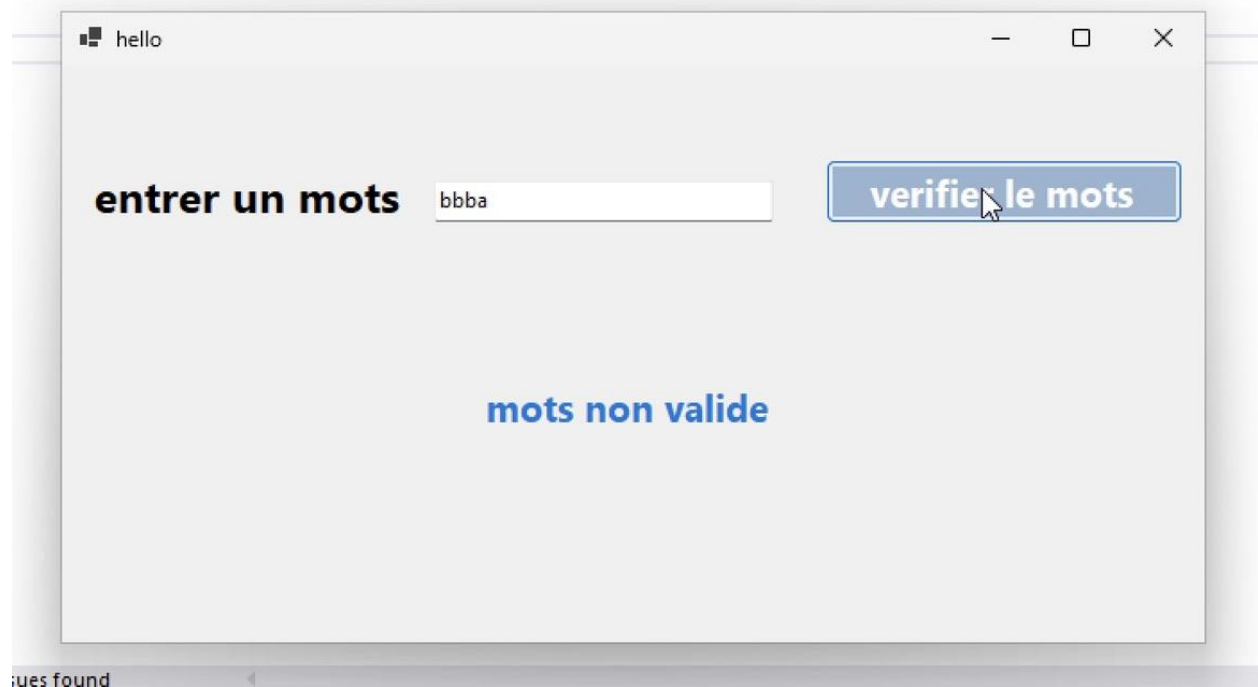
AFD



AFDM



File1.txt*		Form1.cs [Design]	Form1.Design
1	12-1		
2	10-1		
3	0-1-1		



Ex4

$(a+ba)(a+b+c)^*bbb(a+b+c)^*$



The diagram illustrates a sequence of sets of numbers, likely representing a process of building up a set. The sets are:

- $\{1\}$
- $\{1, 2\}$
- $\{1, 2, 3\}$
- $\{1, 2, 3, 4\}$
- $\{1, 2, 3, 4, 5\}$
- $\{1, 2, 3, 4, 5, 6\}$
- $\{1, 2, 3, 4, 5, 6, 7\}$

Arrows indicate a progression from left to right, with some arrows labeled 'a' and 'b'. There are also some additional markings and a small diagram in the top right corner.

