Project Report for Programming

Group B2

- **Project name:** Number Extrapolation

- **Members of our group and their duties:**
    - Basso Nick - Technical Lead & Project Manager
    - Petranis Ann - Project Reviewer and Manager
    - Pascaru Stefan-Alin – Developer of Newton's Interpolation algorithm implementation
    - Chisnicean Ivan - Tester
    - Corman Daniel – Quality Assurance

- **The task of our project** is estimating the 101st number of the set given to us from A groups, using the extrapolation method. We were able to use 3 algorithms: linear, polynomial and conic. Linear extrapolation is to create a tangent line at the end of the known data and to extend it beyond that limit. Polynomial extrapolation is to use Lagrange interpolation or Newton's method of finite differences to create a Newton series that fits the data. Conic extrapolation could be done with a conic sections template, which can be created using five points near the end of the known data. Thus, we decided to implement all this methods to obtain the most exact result.

- **Algorithm**

  Our code begins with the declaration of libraries that are necessary for successful compilation and launch. So, we used `#include <stdio.h>` which stands for "standard input output" header file.

  Our program contains 5 functions for the number extrapolation:

```
void linearTwoPointsExtr();
void linearDeltaSumExtr();
void linearDeltaCoefExtr();
void conicSectionExtr();
void polynomLagrExtr();
```

These functions will be invoked next in main function of the program.

Of course, we used global variables `int numElem` for the number of elements which gave us A groups and `float arr[10000 + 5]` that represents a float-type array of 10000 + 5 elements for the input.

Our program begins with the data input. All necessary values are stored in the file input.txt.

```
// File input declarations
FILE *fp;
fp = fopen("input.txt", "r");

// entering data from the file
int i;

fscanf(fp, "%d", &numElem);

for(i = 0; i < numElem; i++){
	fscanf(fp, "%f", &arr[i]);
}
```

Next starts first function `linearTwoPointsExtr()` - linear extrapolation for two last numbers in the array. In this function we used variables: **y1** - value of the array element at a lower index (lower index = first point index), **longXDelta** - difference between index at which value to be extrapolated and first point index, **shortXDelta** - difference between index at second point and index at first point **yDelta** - difference between array values at first and second point indices and **result** - answer given basing on that function value dependency of argument is linear.

```
void linearTwoPointsExtr(){
	float y1 = arr[numElem - 2];
	float longXDelta = (numElem + 1) - (numElem - 1);
	float shortXDelta = numElem - (numElem - 1);
	float yDelta = arr[numElem - 1] - arr[numElem - 2];
	float result = y1 + longXDelta / shortXDelta * yDelta;

	printf("Linear extrapolation result:\t\t\t\t\t      %f\n", result);
}
```

After that `linearDeltaSumExtr()` function is induced. It fulfills linear delta sum extrapolation. For this function work next variables: **delta** - sum of all differences between two consecutive array elements, **averageDelta** - average of all differences between two consecutive array elements, **result** - averageDelta + last array element value.

```
void linearDeltaSumExtr(){
	int i;
	float delta = 0.0, averageDelta, result;

	for(i = 1; i < numElem; i++){
		delta += arr[i] - arr[i - 1];
	}

	averageDelta = delta / (numElem - 1);

	result = arr[numElem - 1] + averageDelta;

	printf("Linear delta sum extrapolation result:\t\t\t\t	%f\n", result);
}
```

Then program invokes the function of linear delta coefficient extrapolation (`linearDeltaCoefExtr()`). In this function are used variables: **delta** - sum of all coefficients that represent, division of current array element on the previous, **averageDelta** - average of all coefficients described in delta, **result** - last array element value multiplied by averageDelta.

```
void linearDeltaCoefExtr(){
	int i;
	float delta = 0.0, averageDelta, result;

	for(i = 1; i < numElem; i++){
		delta += arr[i] / arr[i - 1];
	}

	averageDelta = delta / (numElem - 1);

	result = arr[numElem - 1] * averageDelta;

	printf("Linear delta average multiplication cofficient extrapolation result:
	%f\n", result);
}
```

Next runs conic section extrapolation function `conicSectionExtr()`. In this function we used following variables: **xt** - stores the sum of all five last x coordinates (which is sum of all array element indices from 1 to number of elements inclusive in this specific task), **yt** - stores the sum of last five numbers given in the input, **xAvg** -

average x-coordinate value obtained from the numbers in xt, **yAvg** - average y-coordinate value obtained from the numbers in yt (five last numbers in the input), **result** - multiplication of x and y coordinate averages multiplied by the number of points taken in examination.

```
void conicSectionExtr(){
	float result, xt = 0, yt = 0;
	float yAvg = 0.0, xAvg = 0.0;
	int i;

	for(i = numElem - 1; i >= numElem - 5; i--){
		xt += i + 1;
		yt += arr[i];
	}

	xAvg = (numElem + 1) / xt;
	yAvg = yt / 5;

	result = yAvg * xAvg * 5;

	printf("Conic section extrapolation result:\t\t\t\t	%f\n", result);
}
```

Finally, our program calls `polynomLagrExtr()` function, which means Lagrange polynomial extrapolation. For this function work next variables: **x** - point, value at which needs to be extrapolated, **k** - Lagrange coefficient obtained by multiplying results between all pairs of x-coordinates where one is obligatory not equal to the other, **result** - equals to the sum of all k-coefficients multiplied by the current y-coordinate value.

```
void polynomLagrExtr(){
//FILE *out1 = fopen("test1.txt", "w");
	float result = 0.0;
	float k;
	float x = numElem + 1;
	int i, j;

	for(j = 0; j < numElem; j++){
		k = 1;
		for(i = 0; i < numElem; i++){
			if(i != j){
				k *= (x - (i + 1)) / ((j + 1) - (i + 1));
				//fprintf(out1, "%d %d %f %f\n", i + 1, j + 1, x, k);
			}
		}
		result += arr[j] * k;
	}
```

```
        printf("Lagrange extrapolation result:\t\t\t\t\t      %f\n", result);
    }
```

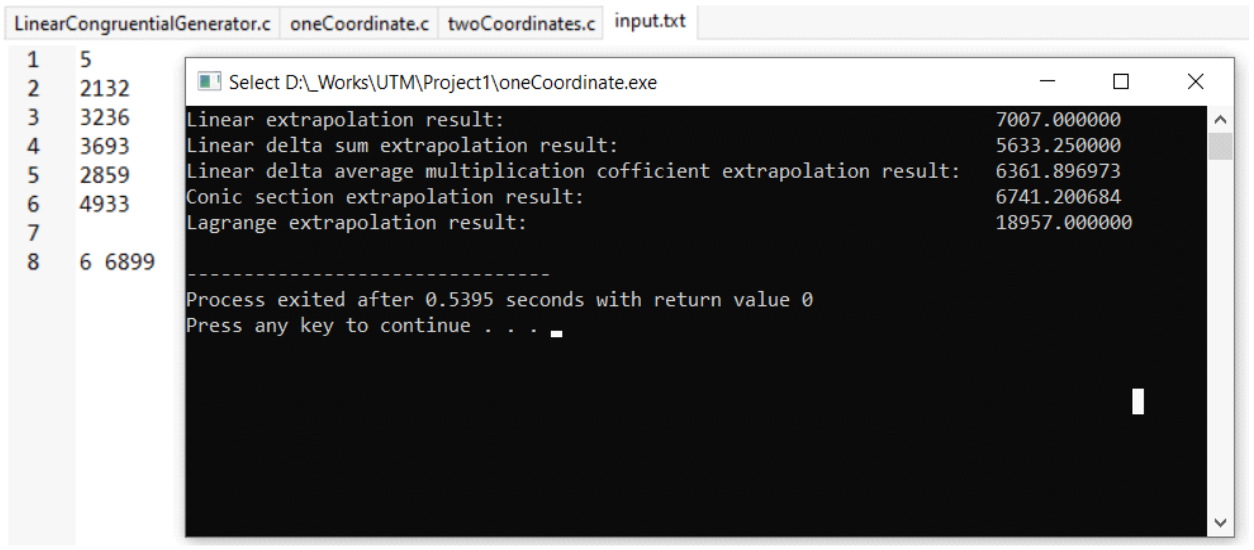
There is also a side task implementation of Newton's interpolation algorithm made by Stefan-Alin Pascaru.

```
void polynomNewtIntr()
{
    float p[10000], f=arr[0], f1=1, f2=0;
    int k=numElem-1,i,j=1;

    do
    {
        for (i=0;i<numElem-1;i++)
        {
            p[i] = ((arr[i+1]-arr[i])/j);
            arr[i]=p[i];
        }
        f1=1;
        for(i=0;i<j;i++)
            {
                f1*=(k-i+1);
            }
        f2+=(arr[0]*f1);
        numElem--;
        j++;
    }
        while(numElem!=1);
    f+=f2;
    printf("Newton Polynomial result:\t\t\t\t\t      %f", f);
}
```

- **Results**

After all these operations we get 5 results. Each result provided may differ from the others because of different extrapolation algorithms used and inputs generated in various ways.


LinearCongruentialGenerator.c
oneCoordinate.c
twoCoordinates.c
input.txt

```
5
2132
3236
3693
2859
4933

6 6899
```

Select D:_Works\UTM\Project1\oneCoordinate.exe

```
Linear extrapolation result:                                     7007.000000
Linear delta sum extrapolation result:                           5633.250000
Linear delta average multiplication cofficient extrapolation result:   6361.896973
Conic section extrapolation result:                              6741.200684
Lagrange extrapolation result:                                   18957.000000

--------------------------------
Process exited after 0.5395 seconds with return value 0
Press any key to continue . . .
```

- Analysis of results