Laborator 2

```c
/* Lab 2  Popa Larisa-Ancuta  Prob 1

-construiti o functie recursiva care calculeaza aranjamente de n luate cate k
-verificati rezultatul folosind si metoda bazata pe factorial

*/

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int aranjamente(int, int); //returneaza nr. de aranjamente de n luate cate k
int factorial(int); //returneaza factorialul unui numar

int main()
{
	int n, k;

	printf("Introduceti doua numere naturale: \n");
	printf("  n=");
	scanf("%d", &n);
	printf("  k=");
	scanf("%d", &k);

	printf("\nAranjamente de n luate cate k sunt: %d\n", aranjamente(n, k));

	if (aranjamente(n, k) == factorial(n) / factorial(n - k))
		printf("\nAcelasi rezultat cu metoda bazata pe factorial. \n\n");
	else
		printf("\nRezultate diferite!\n\n");

	return 0;
}

int aranjamente(int n, int k)
{
	if (k == 1 && n > 0)
		return n;
	else
	{
		if (n == 0 || k == 0)
			return 1;
		else
			return n * aranjamente(n - 1, k - 1);
	}
}

int factorial(int n)
{
	int i, f = 1;

	for (i = 1;i <= n;i++)
		f *= i;

	return f;
}
```

```c
/* Lab 2  Popa Larisa-Ancuta  Prob 2

-calculati combinari de n luate cate k
-verificati rezultatul folosind metoda bazata pe factorial

*/

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int combinari(int, int); //returneaza nr. de combinari de n luate cate k
int factorial(int); //retuneaza factorialul unui numar

int main()
{
	int n, k, f = 0;

	printf("Introduceti doua numere naturale: \n");
	printf("  n=");
	scanf("%d", &n);
	printf("  k=");
	scanf("%d", &k);

	printf("\nCombinari de n luate cate k sunt: %d\n\n", combinari(n, k));

	if (combinari(n, k) == factorial(n) / (factorial(k)*factorial(n - k)))
		printf("Acelasi rezultat cu metoda bazata pe factorial. \n\n");
	else
		printf("\nRezultate diferite!\n\n");

	return 0;
}
int combinari(int n, int k)
{
	if (k == 0)
		return 1;
	else
	{
		if (k > n)
			return 0;
		else
			return combinari(n - 1, k) + combinari(n - 1, k - 1);
	}
}

int factorial(int n)
{
	int i, f = 1;

	for (i = 1;i <= n;i++)
		f *= i;

	return f;
}
```

```c
/* Lab 2  Popa Larisa-Ancuta  Prob 3

-cel mai mare divizor comun a două numere folosind o funcție recursivă

*/

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int cmmdc(int, int); //returneaza cel mai mare divizor comun a doua numere

int main()
{
	int a, b;

	printf("\nIntroduceti doua valori naturale: \n");
	printf("Primul numar: ");
	scanf("%d", &a);
	printf("Al doilea numar: ");
	scanf("%d", &b);

	printf("\nCMMDC este: %d\n\n", cmmdc(a, b));

	return 0;
}

int cmmdc(int a, int b)
{
	if (a == b)
		return a;

	else
		if (a < b)
		    return cmmdc(a, b - a);
	    else
		    return cmmdc(a - b, a);
}
```

```c
/* Lab 2  Popa Larisa-Ancuta  Prob 4

-se consideră recursivitatea (seria de medii arimetico-geometrice a lui Gauss):
    a[n]=(a[n-1]+b[n-1])/2   b[n]=sqrt(a[n-1]*b[n-1])
-determinați a[n] și b[n], pentru n, a[0], b[0] introduse de la tastatură

*/

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <math.h>

int a_0, b_0;

float a_n(int); //returneaza valoarea lui a[n]
float b_n(int); //retruneaza valoarea lui b[n]

int main()
```

```
{
	int n;

	printf("Introduceti trei numere naturale:\n ");
	printf("  n=");
	scanf("%d", &n);
	printf("  a[0]=");
	scanf("%d", &a_0);
	printf("  b[0]=");
	scanf("%d", &b_0);

	printf(" \na[n]= %0.2f", a_n(n));
	printf(" \nb[n]= %0.2f\n\n", b_n(n));

	return 0;
}

float a_n(int n)
{
	if (n == 0)
		return (float)a_0;
	else
		return (float)(a_n(n - 1) + b_n(n - 1)) / 2;
}

float b_n(int n)
{
	if (n == 0)
		return (float)b_0;
	else
		return (float)sqrt(a_n(n - 1)*b_n(n - 1));
}
```

```
/* Lab 2  Popa Larisa-Ancuta  Prob 5

-citiţi un şir de caractere de la tastatură, caracter cu caracter
-afişaţi şirul în ordine inversă folosind o funcţie recursivă

*/

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>

#define DIM 100

void invers(char[DIM], int);//citire caracter cu caracter si inversare sir

int main()
{
	char a[DIM];

	printf("Introduceti caractere(enter=stop):\n ");
	invers(a, 0);

	return 0;
```

```
}

void invers(char a[], int poz)
{
	int i;

	a[poz] = getchar();

	if (a[poz] == '\n')
	{
		printf("\nSirul inversat este:");
		for (i = poz; i >= 0; i--)
			printf("%c", a[i]);
		printf("\n\n");
	}
	else
		invers(a, poz + 1);
}
```

```
/*  Lab 2  Popa Larisa-Ancuta  Prob 6

-determinaţi printr-o funcţie recursivă produsul scalar al doi vectori (aceiasi lungime)

*/

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>

#define DIM 100

void citire(int, int[DIM]); //citire tablou unidimensional
int prod_scalar(int, int[DIM], int[DIM]); //returneaza produsul scalar a doi vectori

int main()
{
	int a[DIM], b[DIM], n;

	printf("Introduceti dimensiunea tablourilor: ");
	scanf("%d", &n);


	printf("\nPrimul vector: \n");
	citire(n, a);

	printf("\nAl doilea vector: \n");
	citire(n, b);

	printf("\n Produsul scalar al celor 2 vectori este: %d\n\n", prod_scalar(n, a,
b));

	return 0;
}

void citire(int n, int x[DIM])
{
```

```
	int i;

	for (i = 0; i < n; i++)
		scanf("%d", &x[i]);
}

int prod_scalar(int n, int a[DIM], int b[DIM])
{
	if (n == 1)
		return a[n - 1] * b[n - 1];
	else
		return prod_scalar(n - 1, a, b) + (a[n - 1] * b[n - 1]);
}
```

```
/* Lab 2  Popa Larisa-Ancuta  Prob 7

-să se calculeze suma numerelor impare dintr-un tablou unidimensional de numere
întregi(recursiv)
-tablou citit dintr-un fișier unde, ca primă valoare, avem numărul de elemente ale
tabloului

*/

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

#define DIM 100

int suma_imp(int, int[DIM]); //returneaza suma numerelor impare din tablou

int main()
{
	char fisier[] = "fisier.txt";
	int a[DIM], n, i;

	FILE *fis;

	if ((fis = fopen(fisier, "r")) == NULL)
		printf("\nFisierul nu a putut fi deschis! \n\n");

	fscanf_s(fis, "%d", &n);

	for (i = 0; i < n; i++)
		fscanf_s(fis, "%d", &a[i]);

	printf("\nSuma numerelor impare este: %d\n\n", suma_imp(n, a));

	fclose(fis);

	return 0;
}

int suma_imp(int n, int a[DIM])
{
	if (n == 0)
		return 0;
```

```
	else
	{
		if (a[n-1] % 2 != 0)
			return a[n-1] + suma_imp(n - 1, a);
		else
			return suma_imp(n - 1, a);
	}
}
```

```
/* Lab 2  Popa Laisa-Ancuta  Prob 8

-se calculează produsul elementelor aflate pe poziţii impare într-un tablou
unidimensional
-să se calculeze suma numerelor prime din tablou

*/

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

#define DIM 100

int produs_poz_imp(int, int[DIM]); //returneaza produsul elementelor de pe pozitii impare
int suma_prim(int, int[DIM]); //returneaza suma elementelor prime din tablou
int prim(int); //verifica daca numarul este prim

int main()
{
	char fisier[] = "fisier.txt";
	int a[DIM], n, i;

	FILE *fis;

	if ((fis = fopen(fisier, "r")) == NULL)
		printf("FIsierul nu a putut fi deschis! \n\n");

	fscanf_s(fis, "%d", &n);

	for (i = 0; i < n; i++)
		fscanf_s(fis, "%d", &a[i]);

	printf("\nProdusul numerelor de pe pozitiile impare este: %d\n", produs_poz_imp(n
- 1, a));
	printf("\nSuma numerelor prime este: %d\n\n", suma_prim(n - 1, a));

	fclose(fis);

	return 0;
}

int produs_poz_imp(int n, int a[DIM])
{
	if (n < 0)
		return 1;
	else
	{
```

```
		if (n % 2 != 0)
			return a[n] * produs_poz_imp(n - 1, a);
		else
			return produs_poz_imp(n - 1, a);
	}
}

int suma_prim(int n, int a[DIM])
{
	if (n < 0)
		return 0;
	else
	{
		if (prim(a[n]))
			return a[n] + suma_prim(n - 1, a);
		else
			return suma_prim(n - 1, a);
	}
}

int prim(int x)
{
	int d;

	if (x == 0)
		return 0;
	else
	{
		if (x == 1 || x == 2)
			return 1;
		else
		{
			for (d = 2; d*d <= x; d++)
				if (x % d == 0)
					return 0;
		}
	}
	return 1;
}
```

```
/*  Lab 2  Popa Larisa-Ancuta  Prob 10

-considerați un șir de n (<=30) de valori întregi
-determinați în mod recursiv și nerecursiv numărul de apariții în șir ale unei valori
întregi x cititite de la tastatură

*/

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

#define DIM 31

int ap_recursiv(int, int[DIM], int); //returneaza nr. de aparitii ale unei valori x
(recursiv)
int ap_nerecursiv(int, int[DIM], int); //returneaza nr. de aparitii ale unei valori x
```

```c
int main()
{
	int a[DIM], n, x, i;

	printf("Introduceti dimensiunea tabloului:");
	scanf("%d", &n);

	printf("Introduceti elementele tabloului: \n");
	for (i = 0; i < n; i++)
		scanf("%d", &a[i]);

	printf("Introduceti o valoare pentru x: ");
	scanf("%d", &x);

	printf("\nValoarea introdusa apare de: \n");
	printf(" -%d ori(recursiv)", ap_recursiv(n - 1, a, x));
	printf("\n -%d ori(nerecursiv)\n\n", ap_nerecursiv(n, a, x));

	return 0;
}


int ap_recursiv(int n, int a[DIM], int x)
{
	if (n < 0)
		return 0;
	else
	{
		if (x == a[n])
			return 1 + ap_recursiv(n - 1, a, x);
		else
			return ap_recursiv(n - 1, a, x);
	}
}

int ap_nerecursiv(int n, int a[DIM], int x)
{
	int nr = 0, i;

	for (i = 0; i < n; i++)
		if (a[i] == x)
			nr++;

	return nr;
}
```

```c
/* Lab 2  Popa Larisa-Ancuta  Prob 11

Considerați un număr n întreg în baza 10 introdus de la tastatură. Folosind o funcție
recursivă
convertiți valoarea n într-o altă bază de numerație 1<b<10 citită de la tastatură

*/

#define _CRT_SECURE_NO_WARNINGS
```

```c
#include <stdio.h>

void conversie(int, int[], int, int);


int main()
{
	int v[33] = { 0 }, b, n, k = 0;

	printf("Dati un numar natural: ");
	scanf("%d", &n);
	printf("Dati baza in care doriti sa convertiti numarul: ");
	scanf("%d", &b);

	conversie(b, v, n, k);

	return 0;
}

void conversie(int b, int v[], int n, int k)
{
	if (n == 0)
		if (k > 4)
			for (int i = k - 1; i >= 0; i--)
				printf("%d", v[i]);
		else
			for (int i = 3; i >= 0; i--)
				printf("%d", v[i]);
	else
	{
		v[k] = n % b;
		conversie(b, v, n / b, k + 1);
	}
}
```

```c
/* Lab 2  Popa Larisa-Ancuta  Prob 12

-fie ecuația de gradul 2: x2 –sx +p=0
-fără a calcula rădăcinile x1 și x2 determinați Sn = x1n +x2n
-folosind reprezentarea recursivă a sumei: Sum(n) ={ 2, dacă n=0; s, dacă n=1; s*Sum(n-1)
– p*Sum(n-2), dacă n>1; }
-verificați dacă e posibil rezultatul obținut

*/

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

float Sum(int, float, float);

int main()
{
	int n;
	float s, p, x, sum_x;

	printf("   x^2 - s*x + p = 0 \n s=");
```

```
	scanf("%f", &s);
	printf(" p=");
	scanf("%f", &p);

	printf("\n Introduceti un numar natural: \n n=");
	scanf("%d", &n);

	printf("\n  Sum(n)= %0.3f\n\n", Sum(n, s, p));

	printf("Verificare\n   x=");
	scanf("%f", &x);

	sum_x = x * x - s * x + p;

	if (Sum(n, s, p) == sum_x)
		printf("\n  Rezultat corect!\n\n");
	else
		printf("\n  Rezultat incorect!\n\n");

	return 0;
}
float Sum(int n, float s, float p)
{
	if (n == 0)
		return 2;
	else
	{
		if (n == 1)
			return s;
		else
			return s * Sum(n - 1, s, p) - p * Sum(n - 2, s, p);
	}
}
```

```
/* Lab 2  Popa Larisa-Ancuta  Prob 13

-scrieți un program care să calculeze valoarea seriei armonice sn=1/1+1/2+1/3+…1/n
-în mod recursiv și în mod nerecursiv
-apelați cele două funcții cu diferite valori ale lui n

*/

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

float seria_ar_rec(int); //returneaza seria armonica recursiv
float seria_ar_nerec(int); //retruneaza seria armonica nerecursiv

int main()
{
	int n;

	printf("Introduceti un numar natural: \n   n=");
	scanf("%d", &n);

	printf("\n(recursiv) Seria armonica este: %0.3f", seria_ar_rec(n));
```

```
	printf("\n(nerecursiv) Seria armonica este: %0.3f\n\n", seria_ar_nerec(n));

	return 0;
}

float seria_ar_rec(int n)
{
	if (n == 1)
		return 1;
	else
		return (float)1 / n + seria_ar_rec(n - 1);
}

float seria_ar_nerec(int n)
{
	int i;
	float s = 0;

	for (i = 1;i <= n;i++)
		s += (float)1 / i;

	return s;
}
```