BALANCE

```
#include <stdio.h>


int main()
{
       int par = 0, ag = 0;
       char c;

       while ((c = getchar() )!= EOF)
       {
              if (c == '(')
                 ag++;
              else if (c == ')')
              ag--;

              if (c == '{')
                 par++;
              else if (c == '}')
              par--;

           if (ag < 0)
           {
              printf("Sfalma stis paren8eseis\n");
              return;
           }
           else if (par < 0)
           {
                printf("Sfalma stis agkules\n");
                return;
           }
     }


     if (ag != 0)
     {
        printf("Sfalma stis paren8eseis\n");
        return;
     }
     else if (par != 0)
     {
          printf("Sfalma stis agkules\n");
          return;
     }
       return 0;
}
```

Άσκηση 2:

ΚΑΡΚΙΝΙΚΗ ΓΡΑΦΗ (odd number of chars)

Παράδειγμα: 1wagaw1

```
#include <stdio.h>

int checkSym(char *name, int L)
{
    int i;
```

```
    if (L % 2 == 0)
       return 0;


    for  (i = 1; i <= L/2; ++i)
    {
         if (name[(L/2)-i] != name[(L/2)+i])
            return 0;
    }

    return 1;

}


int main()
{
       int par = 0, ag = 0;
       char c;
       char name[100];

       scanf("%s",name);

       if (checkSym(name, strlen(name)) == 1)
       {
          printf("%s",name);
       }

       return 0;
}
```

Γράψτε ένα πρόγραμμα που θα διαβάζει ένα τετραγωνικό πίνακα ακεραίων διάστασης N x N που θα έχετε κάνει define και θα υπολογίζει τα αθροίσματα και γινόμενα των γραμμών και των στηλών του. Τέλος θα εκτυπώνει ποια γραμμή και ποια στήλη εμφανίζει το μεγαλύτερο άθροισμα αντίστοιχα.

```
#include <stdio.h>
#define N 3

int getSumRow(int A[N][N],int row)
{
    int j;
    int sum = 0;

    for (j = 0; j < N; ++j)
        sum += A[row][j];

    return sum;
}

int getProdRow(int A[N][N],int row)
{
    int j;
    int prod = 1;

    for (j = 0; j < N; ++j)
        prod *= A[row][j];

    return prod;
}
```

```
int getSumCol(int A[N][N],int col)
{
    int j;
    int sum = 0;

    for (j = 0; j < N; ++j)
        sum += A[j][col];

    return sum;
}

int getProdCol(int A[N][N],int col)
{
    int j;
    int prod = 1;

    for (j = 0; j < N; ++j)
        prod *= A[j][col];

    return prod;
}


int main()
{
       int i,j;
       char c;
       int A[N][N];
       int max = 0,temp;

       for (i = 0; i < N; ++i)
              for (j = 0; j < N; ++j)
                  scanf("%d",&A[i][j]);

       for (i = 0; i < N; ++i)
       {
            temp = getSumRow(A, i);
              if (temp > max)
              {
                     max = getSumRow(A, i);
            }

              printf("Sum-Row[%d] = %d\n",i,getSumRow(A, i));
       }

       printf("Max = %d\n",max);
       max = 0;

       for (i = 0; i < N; ++i)
       {
              temp = getProdRow(A, i);
              if (temp > max)
              {
                     max = getProdRow(A, i);
              }

              printf("Prod-Row[%d] = %d\n",i,getProdRow(A, i));
       }
       printf("Max = %d\n",max);
       max = 0;


       for (i = 0; i < N; ++i)
```

```
	{
		temp = getSumCol(A, i);
		if (temp > max)
		{
			max = getSumCol(A, i);
		}

		printf("Sum-Col[%d] = %d\n",i,getSumCol(A, i));
	}

	printf("Max = %d\n",max);
	max = 0;

	for (i = 0; i < N; ++i)
	{
		temp = getProdCol(A, i);
		if (temp > max)
		{
			max = getProdCol(A, i);
		}

		printf("Prod-Col[%d] = %d\n",i,getProdCol(A, i));
	}

	printf("Max = %d\n ",max);
	return 0;
}
```

Άσκηση 1β

Αν οι p και q είναι και οι δύο πρώτοι και q = p + 2, τότε οι p,q ονομάζονται δίδυμοι πρώτοι, π.χ. το 3 και το 5. Επεκτείνετε το προηγούμενο πρόγραμμα ώστε να τυπώνει όλους τους δίδυμους πρώτους μέχρι LIMIT.

```
#include <stdio.h>
#include <string.h>

#define LIMIT 1000

int main(void)
{
	int j,k;
	int PreviousPrime=2;

	// For every integer k from 2 to LIMIT we check
	// whether there is an integer j that divides it.
	// j runs from 2 to k. When the first divisor j of k has been found,
	// we check whether j==k. If this is the case, then k is a prime.
	// We need to store the last prime found to be able to check
	// for twin primes.
	for (k=2; k < LIMIT; ++k)
	{
		j=2;
		while (k%j != 0)
			j++;

		// If the smallest divisor of k is itself, then k is a prime
		if (j == k)
		{
			// We check whether we have found a pair of
			// twin primes
			if (PreviousPrime+2 == j)
```

```
                        {
                                printf("Found twin primes %d and %d\n",
                                        PreviousPrime, j);
                        }
                        PreviousPrime = j;

                }
        }

        return 0;
}
```

Γράψτε ένα πρόγραμμα που να επιλέγει τυχαία έναν αριθμό από το 1 ως το 100 και να ζητάει από το χρήστη να τον μαντέψει. Αν ο χρήστης μαντέψει τον αριθμό το πρόγραμμα σταματάει, ειδάλλως ανακοινώνει αν ο αριθμός είναι μικρότερος ή μεγαλύτερος από αυτόν που έχει διαλέξει το πρόγραμμα στην αρχή.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define FROM 1
#define TO 100

int IntervalRand(int from, int to)
{
        int res;

        // Obtain a random interval with rand, then linearly scale
        // it to the interval [from, to]

        // rand returns an integer in [0, RAND_MAX]
        // to scale it to [from, to] we use the equation
        // (rand - 0)/(RAND_MAX -0) * (to-from) + from

        res = (((float)rand() * (to-from))/RAND_MAX) + from;
        return res;
}

int main()
{
        int seed, number, guess;

        seed = time(NULL);
        srand(seed);
        number = IntervalRand(FROM, TO);

        printf("My number is %d.\n\n", number);

        printf("I selected a random integer between %d and %d.\n", FROM, TO);
        do
        {
                printf("Enter a number and try to guess it:");
                scanf("%d", &guess);
                if (number == guess)
                {
                        printf("You found it! It's %d\n", number);
                        break;
                }
                if (number < guess)
                        printf("No! My number is smaller than that.\n");
                else
```

```
                printf("No! My number is larger than that.\n");

        } while (1);

        return 0;
}
```

Example: Write your own program that asks user to input sequence of numbers, afterwards it calculates arithmetic middle of the given sequence. Program also prints numbers smaller than arithmetic middle, and afterwards prints numbers bigger than arithmetic middle.

```
#include <stdio.h>
#define DIMENSION 10
int main(void) {
	int i;
	float sum = 0, arit_midd = 0, sequence[DIMENSION]={0};

	for (i = 0; i < DIMENSION; i++)
	{
		printf("Input number: ");
		scanf("%f",&sequence[i]);
		sum += sequence[i];
	}
	arit_midd = sum / DIMENSION;
	printf("Arithmetic middle of the sequence is %6.2f.\n", arit_midd);
	for (i = 0; i < DIMENSION; i++)
		if (sequence[i] < arit_midd)
			printf("%6.2f is smaller than arithmetic middle.\n",
		sequence[i]);

	for (i = 0; i < DIMENSION; i++)
		if (sequence[i] > arit_midd)
			printf("%6.2f is bigger than arithmetic middle.\n",
		sequence[i]);
}
```

Example: Write your own program that asks for input of sequence of numbers. After the program reads given numbers, it divides every number with the biggest sequence element and shows them in a way relative to the biggest element.

```
#include <stdio.h>
#define DIMENSION 10
int main(void)
{
	int i;
	float max, array[DIMENSION];
	for (i = 0; i < DIMENSION; i++)
	{
		printf("array[%d] = ", i);
		scanf("%f",& array[i]);
		if (i == 0)
			max = array[i];
		if (max < array[i])
			max = array[i];
	}
	printf("Biggest element in array is %f.\n\n", max);
	for (i = 0; i < DIMENSION; i++)
	{
		array[i] /= max;
		printf("array[%d] = %f\n", i, array[i]);
	}
}
```

Example: Compose your own program that reads given natural numbers that belong in [10, 99] interval and counts how many times each number showed up. Program stops reading numbers when

element that doesn't belong to interval is given. Afterwards, program prints each number from the interval that has showed at least once, and number of times it has really been given.

```
#include <stdio.h>
#define LL 10 /* lower limit of the interval */
#define UL 99 /* upper limit of the interval */
int main(void)
{
	int number, i;
	int counter[UL - LL + 1] = { 0 };
	do {
		printf("\nInput number from interval [%d, %d]: ", LL, UL);
		scanf("%d", &number);
		if (number >= LL && number <= UL)
			counter[number - LL]++;
	} while (number >= LL && number <= UL);

	for (i = DG; i <= UL; i++)
		if (counter[i - LL] > 0) {
			printf("\nNumber %d showed up %d times", i, counter [i -
				LL]);

	}
}
```

Example : Write your own C program that transposes matrix. Program stores given matrix dimensions and every single matrix element must be given. Transposed matrix is the one with rows and columns switched.

```
#include <stdio.h>
#define MAX_ROW 50
#define MAX_COL 50
int main(void)
{
	int i, j, m, n, temp;
	int mat[MAX_ROW][MAX_COL];
	int dim = (MAX_ROW < MAX_COL)? MAX_ROW : MAX_COL;
	do {
		printf("Input number of rows < %d :", dim);
		scanf("%d", &m);
		printf("Input number of columns < %d:", dim);
		scanf("%d", &n);
	} while (m < 1 || m > dim || n < 1 || n > dim);
	printf("\nInput of matrix elements :\n");

	for (i = 0; i < m; i++)
		for (j = 0; j < n; j++)
		{
			printf("Input element [%d][%d] : ", i, j);
			scanf("%d", &mat[i][j]);
		}

	printf("\n\nMatrix before transposing:\n");
	for (i = 0; i < m; i++)
	{
		for (j = 0; j < n; j++)
			printf("%3d", mat[i][j]);
		printf("\n");
	}
```

```
        for ( i=0; i<m; ++i )
                for ( j=i+1; j<n; ++j )
                {
                        temp = mat[i][j];
                        mat[i][j] = mat[j][i];
                        mat[j][i] = temp;
                }

        printf("\nMatrix after transposing:\n");
        for (i = 0; i < n; i++) {
                for (j = 0; j < m; j++)
                        printf("%3d", mat[i][j]);
                printf("\n");
        }
}
```

Multiplication

```
 1  void    mm_mul (int A[N][N], int B[N][N], int C[N][N])
 2  {
 3          int i, j, k;
 4          int sum;
 5
 6          for (i = 0; i < N; i++) {
 7                  for (j = 0; j < N; j++) {
 8                          sum = 0;
 9                          for (k = 0; k < N; k++) {
10                                  sum += A[i][k] * B[k][j];
11                          }
12                          C[i][j] = sum;
13                  }
14          }
15  }
```