- Αναπτύξτε πρόγραμμα που με χρήση της συνάρτησης average(int numbers [], int num) να υπολογίζει τον αριθμητικό μέσο των θετικών στοιχείων μία μήτρας ακεραίων αριθμών.

```
#include <iostream>
using namespace std;

double average(int numbers[],int num)
{
	int posElements = 0;
	double sum = 0;
	double average;

	for(int i = 0;i < num;i++)
	{
		int element = numbers[i];
		if(element > 0)
		{
			posElements++;
			sum += element;
		}
	}

	average = sum /posElements;
	return average;
}

int main()
{
	int nums[] = {-5,4,9,-10,-100,1000,1};
	cout << average(nums,7);
	return 0;
}
```

```
#include <iostream>
using namespace std;

double average(int numbers[],int num)
{
      …..declarations

      for(int i = 0;i < num;i++)
      {









      }

      average =…..
      return ….;
}

int main()
{
      int nums[] = {-5,4,9,-10,-100,1000,1};
      cout  …..;
      return ……;
}
```

- Αναπτύξτε πρόγραμμα εύρεσης ελάχιστου αριθμού από δισ-διάστατη μήτρα ακεραίων αριθμών που θα εισάγεται από το πληκτρολόγιο.

```
#include <iostream>

using namespace std;
int main ()
{
    int rows, columns;
    do
    {
        cout << " Give the number of rows "  << " and columns ";
        cin >> rows >> columns;
    }
    while ((rows<columns?rows:columns)<=0);
    int x[....;

    for (int i=0; i<rows; i++)
        for (int j=0; j<columns; j++)
        {
                .....
        }
    cout << " Minimum element = " << min << endl;
    return 0;
 }
```

```
#include <iostream>

using namespace std;
int main ()
{
    int rows,columns,min=32768;
    do
    {
        cout << " Give the number of rows "  << " and columns ";
        cin >> rows >> columns;
    }
    while ((rows<columns?rows:columns)<=0);

    int x[rows][columns];

    for (int i=0; i<rows; i++)
        for (int j=0; j<columns; j++)
        {
            cout <<" Give number x["<<i<<"]["<<j<<"]"<< endl;
            cin >> x[i][j];
                if (x[i][j]<min) min=x[i][j];
        }
    cout << " Minimum element = " << min << endl;
    return 0;
}
```

- Αναπτύξτε βελτιστοποιημένο πρόγραμμα εύρεσης ελάχιστου αριθμού από δισ-διάστατη μήτρα ακεραίων αριθμών που θα εισάγεται από το πληκτρολόγιο.

```
#include <iostream>
#include <limits.h>

using namespace std;
int main ()
{
      // an optimized version
      short rows,columns;
      int x, min = INT_MAX;
      do
      {
                cout << " Give the number of rows "
       << " and columns==>";
                cin >> rows >> columns;
      }
      while ((rows<columns?rows:columns)<=0);
      for(int register i=0; i<rows; i++)
                for(int register j=0; j<columns; j++)
                {
                cout << " Value of a["<<i<<','<<j<<"]=";
                          cin >> x;
                          if(x<min) min=x;
                }
      cout << " Minimum element = " << min << endl;
      return 0;
 }
```

- Αναπτύξτε βελτιστοποιημένο πρόγραμμα εύρεσης μεγίστου από τα ελάχιστα των γραμμών μήτρας ακεραίων αριθμών που θα εισάγεται από το πληκτρολόγιο.

```
#include <iostream>
#include <limits.h>
using namespace std;

int main()
{
	short rows, columns;
	int x;
	do
	{
		cout << "Dwse arithmo grammwn kai steilwn" << endl;
		cin >> rows >> columns;
	} while ((rows<columns?rows:columns)<=0);
	int i, j;
	int register min, row_max=INT_MIN;
	for (i=0; i<rows; i++)
	{
		min=INT_MAX;
		for (j=0; j<columns; j++)
		{
			cout << "Dwse timi gia a[" << i <<"," << j << "]:";
			cin >> x;
			min= x<min? x:min;
		}
		row_max=row_max>min? row_max:min;
	}
	cout << "Maximun of minima =" << row_max << endl;
	return 0;
}
```

- Αναπτύξτε βελτιστοποιημένο πρόγραμμα εύρεσης ελάχιστου μη αρνητικού αριθμού από μήτρα ακεραίων αριθμών που θα εισάγεται από το πληκτρολόγιο.

```
#include <iostream>
#include <limits.h>
using namespace std;
int main ()
{
	// an optimized version
	short rows,columns;
	int x, min = INT_MAX;
	do
	{
		cout << " Give the number of rows "
		 << "and columns==>";
		cin >> rows >> columns;
	}
	while ((rows<columns?rows:columns)<=0);
	for(int register i=0; i<rows; i++)
		for(int register j=0; j<columns; j++)
		{
			cout << " Value of a["<<i<<','<<j<<"]=";
			cin >> x;
			if(x <=0) continue;
			if(x<min) min=x;
		}
		cout << " Minimum non negative element = " << min << endl;
	return 0;
}
```

- Να δημιουργήσετε μια εφαρμογή στη C++ στην οποία θα κατασκευάσετε τρεις συναρτήσεις
  - την ReadPin, η οποία δέχεται ένα πίνακα ακεραίων και το πλήθος των στοιχείων του και γεμίζει το πίνακα με τιμές από το πληκτρολόγιο.
  - την ShowPin, η οποία δέχεται ένα πίνακα ακεραίων και το πλήθος των στοιχείων του και εμφανίζει τις τιμές του πίνακα στην οθόνη του υπολογιστή
  - την Average, η οποία δέχεται ένα πίνακα ακεραίων και το πλήθος των στοιχείων του και επιστρέφει το μέσο όρο των τιμών του πίνακα
- Εν συνεχεία να υπερφορτώσετε τις ανωτέρω τρεις συναρτήσεις ώστε να λειτουργούν για πίνακες πραγματικών αριθμών.
- Μέσα στην main να δηλώσετε το πίνακα ακεραίων iPin με πλήθος 3 και το πίνακα πραγματικών fPin με πλήθος 5 στοιχεία. Με τη χρήση των ανωτέρω συναρτήσεων και για τους δύο πίνακες να τους γεμίσετε με τιμές από το πληκτρολόγιο, να εμφανίσετε τις τιμές, να εμφανίσετε το μέσο όρο τους.

```
#include <iostream>
using namespace std;

void ReadPin(float *Pin, int n)
{
    for (int i = 0; i < n; i++)
    {
        cout << "Type a float : ";
        cin >> Pin[i];
    }
}

void ReadPin(int *Pin, int n)
{
    for (int i = 0; i < n; i++)
    {
        cout << "Type an integer : ";
        cin >> Pin[i];
    }
}
```

```
void ShowPin(float *Pin, int n)
    {
        for (int i = 0; i < n; i++)
        {
            cout << Pin[i] ;
        }
        cout << endl;
    }

void ShowPin(int *Pin, int n)
    {
        for (int i = 0; i < n; i++)
        {
            cout << Pin[i] ;
        }
        cout << endl;
    }
```

```
int Average(int *Pin, int n)
    {
      int sum = 0;

        for (int i = 0; i < n; i++)
        {
            sum = sum + Pin[i];
        }
        return sum / n;
    }
float Average(float *Pin, int n)
    {
      float sum = 0.0;

        for (int i = 0; i < n; i++)
        {
            sum = sum + Pin[i];
        }
        return sum / n;
    }
```

```
int main()
    {
        int iPin[3];
        float fPin[5];

        ReadPin(iPin,3);
        ShowPin(iPin,3);
        cout << Average(iPin, 3) <<
    endl;

        ReadPin(fPin,5);
        ShowPin(fPin,5);
        cout << Average(fPin, 5) <<
    endl;

        return 0;
    }
```

- Να δημιουργήσετε τη κλάση Rectangle η οποία έχει σαν ιδιωτικά μέλη τις δύο πλευρές του ορθογωνίου παραλληλογράμμου και σαν δημόσια μέλη τις μεθόδους :
  - Embadon η οποία υπολογίζει και επιστρέφει το εμβαδόν του παραλληλογράμμου (πλευρά A * πλευρά B)
  - Perimetros η οποία υπολογίζει και επιστρέφει τη περίμετρο του παραλληλογράμμου 2 * (πλευρά A + πλευρά B)
  - getA η οποία επιστρέφει τη τιμή της πλευράς A
  - getB η οποία επιστρέφει τη τιμή της πλευράς B
  - setA η οποία αποδίδει στη πλευρά A μια νέα τιμή μόνο αν αυτή είναι μεγαλύτερη από 0. Επίσης επιστρέφει τη τιμή true αν η απόδοση είναι σωστή διαφορετικά τη τιμή false.
  - setB η οποία αποδίδει στη πλευρά B μια νέα τιμή μόνο αν αυτή είναι μεγαλύτερη από 0. Επίσης επιστρέφει τη τιμή true αν η απόδοση είναι σωστή διαφορετικά τη τιμή false.
  - scale η οποία δέχεται ένα αριθμό και κλιμακώνει τις δύο πλευρές. Π.χ. αν ο αριθμός είναι το 2 τότε διπλασιάζει τις δύο πλευρές.
  - scale η οποία δέχεται δύο αριθμούς-ορίσματα και κλιμακώνει τη πρώτη πλευρά κατά το πρώτο όρισμα και τή δεύτερη πλευρά κατά το δεύτερο όρισμα. Π.χ. αν οι αριθμοί είναι το 2 και το 0.5 τότε διπλασιάζει τη πρώτη πλευρά και μειώνει στο μισό τη δεύτερη πλευρά.
  - print η οποία δέχεται ένα ακέραιο αριθμό και αν είναι 1 τότε εκτυπώνει στη οθόνη το εμβαδόν, αν είναι 2 εκτυπώνει τη περίμετρο και αν είναι οτιδήποτε άλλο εκτυπώνει και το εμβαδόν και τη περίμετρο.
- Επίσης να δημιουργήσετε ένα κατασκευαστή της κλάσης, ο οποίος θα δέχεται τις τιμές των δύο πλευρών και θα έχει σαν προκαθορισμένες τις τιμές 2 και 4 αντίστοιχα. Μέσα στο κατασκευαστή δεν θα γίνεται έλεγχος των τιμών αν είναι μεγαλύτερες από 0.

```
#include <iostream>
using namespace std;
class Rectangle
    {
        private :
           float plA, plB;

        public :

        Rectangle(float a = 2.0, float b=4.0)
        {
           plA = a;
           plB = b;
        }

        float Embadon()
        {
            return plA * plB;
        }

        float Perimetros()
        {
            return 2.0 * (plA + plB);
        }
bool setPleyraA(float a)
    {
        if(a > 0.0)
        {
           plA = a;
           return true;
        }
        return false;
    }
bool setPleyraB(float b)
    {
        if(b > 0.0)
        {
           plB = b;
           return true;
        }
        return false;
    }
float getA(void)
    {
          return plA;
    }
float getB(void)
    {
          return plB;
    }
```

```
void scale(float factor)
    {
       plA = factor * plA;
       plB = factor * plB;
    }

void scale(float factorA, float factorB)
    {
       plA = factorA * plA;
       plB = factorB * plB;
    }

 void print(int option)
    {
        if(option == 1)
        {
          cout << "Embadon " << Embadon() << endl;
        }
        else if(option == 2)
        {
          cout << "Perimetros " << Perimetros() <<
      endl;
        }
        else
        {
           cout << "Embadon " << Embadon() << endl;
           cout << "Perimetros " << Perimetros() <<
      endl;
        }
    }
};
```

- Να κατασκευάσετε μια εφαρμογή όπου :
  - 1) θα δημιουργήσετε 2 αντικείμενα της κλάσης Rectangle, ένα με το προκαθορισμένο κατασκευαστή (χωρίς παραμέτρους) και ένα αντικείμενο με τιμές 4 και 6.
  - 2) Να υπολογίσετε και να εμφανίσετε το εμβαδόν και τη περίμετρο των δύο αντικειμένων.
  - 3) Κλιμακώστε το πρώτο αντικείμενο με τη τιμή 3 και το δεύτερο με τις τιμές 3 και 5.
  - 4) Να υπολογίσετε και να εμφανίσετε το εμβαδόν και τη περίμετρο των δύο αντικειμένων.

```
int main()
{
    Rectangle t1;
    Rectangle t2(4.0,6.0);

    t1.print(3);
    t2.print(3);
    t1.scale(2.0);
    t2.scale(3.0, 4.0);
    t1.print(3);
    t2.print(3);

     return 0;
}
```

- Να δημιουργήσετε τη κλάση OneDate η όποια θα έχει τη λειτουργικότητα για τη διαχείριση μιας ημερομηνίας. Θα έχει σαν ιδιωτικά μέλη :
  - 1. Τρεις ακεραίους αριθμούς οι οποίοι θα αντιπροσωπεύουν την ημέρα, το μήνα και το χρόνο μιας ημερομηνίας.
  - 2. Έναν ακέραιο αριθμό, ο οποίος θα αντιπροσωπεύει την μορφή της ημερομηνίας ως εξής : 0 – Ελληνικά (H/M/XP) και 1 – Αγγλικά (M/H/XP)
- Θα έχει σαν δημόσια μέλη :
  - 1. Μια συνάρτηση κατασκευαστή, οι οποία θα δίνει τιμές σε όλα τα ιδιωτικά μέλη και θα έχει σαν προκαθορισμένες τιμές : 1 για την ημέρα, 1 για το μήνα, 2000 για το χρόνο και 0 που σημαίνει ότι είναι Ελληνικά.
  - 2. Να δημιουργήσετε 2 μεθόδους για την απόδοση (setDay) και την επιστροφή της τιμής (getDay) της ιδιότητας day.
  - 3 . Να γίνετε έλεγχος στη setDay αν η αποδιδόμενη τιμή στην ιδιότητα day, είναι μικρότερη η ίση με το 0 ή μεγαλύτερη από 31 και αν ναι, τότε δεν θα γίνεται απόδοση της τιμής.
  - 4. Να δημιουργήσετε 2 μεθόδους για την απόδοση (setMonth) και την επιστροφή της τιμής (getMonth) της ιδιότητας month.
  - 5. Να γίνετε έλεγχος στη setMonth αν η αποδιδόμενη τιμή είναι μικρότερη η ίση με το 0 ή μεγαλύτερη από 12 και αν ναι τότε δεν θα γίνεται απόδοση της τιμής.
  - 6 . Να δημιουργήσετε 2 μεθόδους για την απόδοση (setYear) και την επιστροφή της τιμής (getYear) της ιδιότητας year.

- 7 . Να γίνετε έλεγχος στη setYear αν η αποδιδόμενη τιμή είναι μικρότερη η ίση με το 0 και σε αυτή τη περίπτωση δεν θα γίνεται απόδοση της τιμής.
- 8 . Να δημιουργήσετε 2 μεθόδους για την απόδοση (setLocale) και την επιστροφή της τιμής (getLocale) της ιδιότητας Locale.
- 9 . Να γίνετε έλεγχος στη setLocale αν η αποδιδόμενη τιμή είναι διάφορη από το 0 ή 1 και σε αυτή τη περίπτωση δεν θα γίνεται απόδοση της τιμής.
- 10. Να δημιουργήσετε μια μέθοδο με το όνομα showDate για την εμφάνιση της ημερομηνίας ανάλογα με το locale, δηλαδή αν είναι 0 τότε θα εμφανίζεται day/month/year ενώ αν είναι 1 τότε θα εμφανίζεται month/day/year.
- 11 . Να δημιουργήσετε μια μέθοδο Compare η οποία θα δέχεται τα στοιχεία μιας ημερομηνίας HM1 (ημέρα, μήνα, χρόνος) και θα τα συγκρίνει με τα ιδιωτικά μέλη της κλάσης και θα επιστρέφει 0 αν είναι ίδιες οι ημερομηνίες, 1 η ημερομηνία HM1 είναι μετά από τη ημερομηνία των ιδιωτικών μελών, και -1 στην αντίθετη περίπτωση .
  - Π.χ. αν η ημερομηνία της κλάσης είναι 1/1/2000 και συγκρίνω :
    - Α) με την ημερομηνία 1/1/2000 τότε επιστρέφει 0
    - Β) με την ημερομηνία 2/1/2000 τότε επιστρέφει 1
    - Γ) με την ημερομηνία 1/1/1999 τότε επιστρέφει -1
- 12 . Να υπερφορτώσετε τη μέθοδο Compare η οποία θα δέχεται μια ημερομηνία σαν κείμενο π.χ «12/10/2010» και ένα ακέραιο αριθμό που θα αντιστοιχεί στο locale και θα την συγκρίνει με τα ιδιωτικά μέλη της κλάσης και θα επιστρέφει 0 αν είναι ίδιες οι ημερομηνίες, 1 η ημερομηνία HM1 είναι μετά από τη ημερομηνία των ιδιωτικών μελών, και -1 στην αντίθετη περίπτωση .
- 13 . Να υπερφορτώσετε τη συνάρτηση κατασκευαστή ώστε να δέχεται μια ημερομηνία σαν κείμενο π.χ «12/10/2010» και ένα ακέραιο αριθμό που θα αντιστοιχεί στο locale και θα αποδίδει τις τιμές στα ιδιωτικά μέλη της.

```
#include <iostream>
using namespace std;
class OneDate
{
  public :
  int day, month, year;
  int locale;
  public:

  OneDate(int x = 1, int y = 1, int z = 2000, int
     locale = 0)
  {
     day = x;
     month = y;
     year = z;
  }

  void setDay(int x)
  {
      if((x > 0) && (x < 32))
      {
         day = x;
      }
  }

  int getDay()
  {
      return day;
  }

  void setMonth(int x)
  {
      if((x > 0) && (x < 13))
      {
        month = x;
      }
  }

  int getMonth()
  {
      return month;
  }

  void setYear(int x)
  {
    if(x > 0)
    {
      year = x;
    }
  }

  int getYear()
  {
      return year;
  }
```

```
void setLocale(int x)
  {
     if((x == 0) || (x == 1))
     {
          locale = x;
     }
  }
  int getLocale()
  {
     return locale;
  }
  void showDate()
  {
      if(locale == 0)
      {
        cout << day << "/" << month << "/" <<
     year << endl;
      }
      else
      {
        cout << month << "/" << day << "/" <<
     year << endl;
      }
  }
```

```
int Compare(int x, int y, int z)
  {
     int t1 = (z * 365) + (y * 31) + x;
     int t2 = (year * 365) + (month * 31) + day;

     if(t1 > t2)
     {
          return 1;
     }
     else if(t1 < t2)
     {
         return -1;
     }
     else
     {
        return 0;
     }
  }
```

```
char *substring(char *src, char *tmp, int start, int end)
 {
     int k = 0;

     for(int i=start; i < end; i++)
     {
        tmp[k++] = src[i];
     }
     tmp[k] = '\0';
     return tmp;
 }

 int Compare(char *src, int locale)
 {
     int xday, xmonth, xyear;
     char dst[5];

     xday = atoi(substring(src, dst, 0,2));
     xmonth = atoi(substring(src,dst,3,5));
     xyear = atoi(substring(src,dst, 6,10));
     int tmp;

     if(locale == 1)
     {
       return Compare(xmonth, xday, xyear);
     }
     else
     {
       return Compare(xday, xmonth, xyear);
     }
 }
```

```
OneDate(char *s, int xlocale)
    {
      char dst[5];
      day = atoi(substring(s, dst, 0,2));
      month = atoi(substring(s, dst, 3,5));
      year = atoi(substring(s,dst,6, 10));
      locale = xlocale;
      int tmp;

      if(locale == 1)
      {
        tmp = day;
        day = month;
        month = tmp;
      }
    }

};
```

```
int main()
{
   OneDate x("09/12/2010",0);
   OneDate z("12/12/2010",0);
   if(x.Compare(z.getDay(), z.getMonth(), z.getYear()) < 0)
   {
      cout << "Megalyteri einai : ";
      x.showDate();
   }
   else if(x.Compare(z.getDay(), z.getMonth(), z.getYear()) > 0)
   {
      cout << "Megalyteri einai : ";
      z.showDate();
   }
   else
   {
        cout << "Einai idies. " << endl;
   }
   return 0;
}
```