Άσκηση 1

Να γραφεί συνάρτηση με όνομα median() που να δέχεται στην είσοδό της έναν πίνακα ακεραίων μαζί με το πλήθος τους, να τους ταξινομεί σε αύξουσα διάταξη και να επιστρέφει τον διάμεσο αριθμό. Αν το πλήθος των αριθμών είναι περιττό (N = 2k+1) τότε ο διάμεσος θα είναι στην (k+1) θέση ενώ αν είναι άρτιο (N = 2k) τότε ο διάμεσος είναι στη θέση k.

Παράδειγμα: 29 6 9 20 2 20 23 --> 2 6 9 (20) 20 23 29

διάμεσος

7 9 1 18 9 9 --> 1 7 (9) 9 9 18

```
int median(int a[], int N)
{
  int i, j, temp;

  for(i=0; i <N-1; i++)   /* Sort array */
    for(j=i+1; j<N; j++)
      if(a[i] > a[j])
      {
        temp = a[i];
        a[i] = a[j];
        a[j] = temp;
      }

    return a[(N-1)/2];
}
```

Άσκηση 2

Δημιουργήστε μία συνάρτηση που να δέχεται σαν παραμέτρους δύο πίνακες ακεραίων και τον αριθμό (π.χ. N) των στοιχείων που επιθυμούμε να συγκριθούν. Αν τα N πρώτα στοιχεία των δύο πινάκων έχουν τις ίδιες τιμές, η συνάρτηση να επιστρέφει 1, αλλιώς 0.

```
int compare(int A[], int B[], int N)  /* 1os tropos */
{
  int i, same = 1;

  for(i=0; i<N; i++)  /* also:  for(i=0; i<N && same; i++) */
    if(A[i] != B[i])
      same = 0;

  return same;
}


int compare(int A[], int B[], int N)  /* 2os tropos */
{
  int i;

  for(i=0; i<N; i++)
    if(A[i] != B[i])
      return 0;           /* amesh ejodos! */

  return 1;
}
```

Άσκηση 3

Να γράψετε ένα πρόγραμμα το οποίο να εμφανίζει όλους τους πρώτους αριθμούς p που βρίσκονται ανάμεσα από δύο ακεραίους n_1 και n_2, δηλαδή όλους τους p: $n_1 < p < n_2$. Υπενθυμίζουμε ότι πρώτος είναι ένας ακέραιος αριθμός μεγαλύτερος του 1 ο οποίος διαιρείται μόνο από το 1 και τον εαυτό του. Παράδειγμα πρώτων αριθμών: 2, 3, 5, 7, 11, 13, 17, 19, 23, . . .

```
#include <stdio.h>

int prime_num(int p);

int main()
{
  int   i, n1, n2;

  printf("Enter 2 positive numbers: ");
  scanf("%d%d",&n1,&n2);
  printf("\nPrime numbers in the interval (%d, %d): ",n1,n2);

  for(i=n1+1; i<n2; i++)
    if(prime_num(i))
      printf("%d ",i);

  return 0;
}
```

```
/* Λογική συνάρτηση για τον έλεγχο αν ένας αριθμός είναι πρώτος */

int prime_num(int p)
{
  int   i=2, flag=1;

  while(i <= sqrt(p) && flag)
  {
    if(p%i == 0)
      flag = 0;
    i++;
  }
  return flag;
}
```

ή

```
int prime_num(int p)
{
  int   i=2;

  while(i <= sqrt(p))
  {
    if(p%i == 0)
      return 0;
    i++;
  }
  return 1;
}
```

Άσκηση 3 (συνέχεια)

Να ξαναγραφεί το προηγούμενο πρόγραμμα ώστε να εμφανίζει τους πρώτους αριθμούς που το πρώτο και τελευταίο ψηφίο τους είναι ίδια. Ο έλεγχος για τα ψηφία να γίνεται από άλλη λογική συνάρτηση.

```
#include <stdio.h>

int prime_num(int p);
int first_eq_last(int num);

int main()
{
  int   i, n1, n2;

  printf("Enter 2 positive numbers: ");
  scanf("%d%d",&n1,&n2);
  printf("\nPrime numbers in the interval (%d, %d): ",n1,n2);


  for(i=n1+1; i<n2; i++)
    if(prime_num(i) && first_eq_last(i))
      printf("%d ",i);


  return 0;
}
```

```
/* Λογική συνάρτηση για τον έλεγχο αν ένας αριθμός είναι πρώτος */

int prime_num(int p)
{
  int   i=2;

  while(i <= sqrt(p))
  {
    if(p%i == 0)
      return 0;
    i++;
  }
  return 1;
}

/* Λογική συνάρτηση για τον έλεγχο πρώτου και τελευταίου ψηφίου */

int first_eq_last(int num)
{
  int   last;

  last = num%10;
  while(num/10 > 0) /* at the end of loop first digit is in num */
    num = num/10;
  if (last != num)
    return 0;
  else
    return 1;
}
```

Άσκηση 3 (συνέχεια)

Να γράψετε μια αναδρομική συνάρτηση που να ελέγχει αν ένας αριθμός είναι δύναμη του δύο. Στη συνέχεια, να τροποποιήσετε το προηγούμενο πρόγραμμα ώστε να εμφανίζει τους πρώτους που είτε είναι κατά ένα μικρότεροι ή κατά ένα μεγαλύτεροι από κάποια δύναμη του 2.

```
int pow_2(int num)
{
  if(num == 1)
    return 1;
  else if(num%2 != 0)
    return 0;
  else
    return pow_2(num/2);
}
```

ή

```
int pow_2(int num)
{
  if(num == 1)
    return 1;
  if(num%2 != 0)
    return 0;
  else
    return pow_2(num/2);
}
```

```
int main()
{
  . . .

  for(i=n1+1; i<n2; i++)
    if(prime_num(i) && (pow_2(i+1) || pow_2(i-1)))
      printf("%d ",i);
  . . .
}
```

Άσκηση 4

α) Να γράψετε μια συνάρτηση με όνομα **isvowel** που να δέχεται στην είσοδό της έναν χαρακτήρα και να επιστρέφει την τιμή 1 αν αυτός αντιστοιχεί σε φωνήεν. Σε αντίθετη περίπτωση, να επιστρέφει την τιμή 0.

```
int isvowel(char ch)  /* using switch */
{
  switch (ch)
  {
    case 'a':
    case 'e':
    case 'i':
    case 'o':
    case 'u': return 1;
    default:  return 0;
  }
}

int isvowel(char ch)  /* using if */
{
  if(ch=='a' || ch=='e' || ch=='i' || ch=='o' || ch=='u')
    return 1;
  else
    return 0;
}
```

Άσκηση 4 (συνέχεια)

β) Να γράψετε ένα πρόγραμμα που να προτρέπει τον χρήστη να πληκτρολογήσει μια γραμμή με αλφαριθμητικούς χαρακτήρες και να εμφανίζει στην οθόνη το πλήθος των φωνηέντων. Το πρόγραμμα να χρησιμοποιεί τη συνάρτηση **isvowel** του προηγούμενου ερωτήματος.

```
#include <stdio.h>

int isvowel(char ch);

int main()
{
  int  cnt=0;
  char ch;

  printf("Please enter a sentence\n");

  while ((ch = getchar()) != '\n')
    cnt += isvowel(ch);

  printf("There were %d vowels in the sentence.\n",cnt);

  return 0;
}
```

Άσκηση 5

Γράψτε μια συνάρτηση με δύο παραμέτρους, τους ακεραίους position και capital. Η συνάρτηση θα επιστρέφει το γράμμα που βρίσκεται στην αντίστοιχη θέση position του λατινικού αλφαβήτου και συγκεκριμένα αν το capital είναι 1 θα επιστρέφει το κεφαλαίο ενώ αν είναι 0 θα επιστρέφει το αντίστοιχο πεζό γράμμα. Στην περίπτωση που κάποια παράμετρος είναι εκτός ορίων η συνάρτηση θα επιστρέφει το χαρακτήρα '\0'. Π.χ.

Παράμετροι		Η συνάρτηση επιστρέφει:
position	capital	
4	1	D
4	0	d
26	1	Z
27	0	\0
3	2	\0
-4	1	\0

```
char  gramma(int position, int capital)
{
  if((position<=0) || (position>26) || ((capital!=0) && (capital!=1)))
    return '\0';

  return ((capital == 1) ? position+64 : position+96);
  /* return ((capital == 1) ? position-1+'A' : position-1+'a'); */
}
```

Άσκηση 6

Να δημιουργήσετε τις συναρτήσεις f() και g(), σύμφωνα με τους παρακάτω τύπους:

$$f(x)=\begin{cases} x+2, & x>0 \\ -3x+7, & x\le 0 \end{cases} \qquad g(x)=\begin{cases} x^2+2, & x>0 \\ 7x-5, & x\le 0 \end{cases}$$

Να γραφεί ένα πρόγραμμα το οποίο να διαβάζει έναν ακέραιο (π.χ. a) και να χρησιμοποιεί τις συναρτήσεις f() και g() για να εμφανίσει το αποτέλεσμα της παράστασης f(g(a)).

```
#include <stdio.h>

int f(int x);
int g(int x);

int main(void)
{
  int a;

  printf("Enter number: ");
  scanf("%d", &a);

  printf("%d\n", f(g(a)));
  return 0;
}
```

```
int f(int x)
{
  if(x > 0)
    return x+2;
  else
    return -3*x+7;
}

int g(int x)
{
  if(x > 0)
    return x*x+2;
  else
    return 7*x-5;
}
```