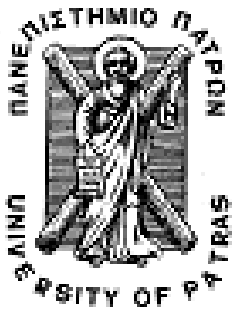
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ & ΠΛΗΡΟΦΟΡΙΚΗΣ



ΕΙΣΑΓΩΓΗ ΣΤΟ ΔΙΑΔΙΚΑΣΤΙΚΟ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ (2009-2010)

ΥΠΕΥΘΥΝΟΙ ΔΙΔΑΣΚΟΝΤΕΣ ΕΡΓΑΣΤΗΡΙΟΥ: Α. ΦΩΚΑ, Κ. ΣΤΑΜΟΣ

4ο ΣΕΤ ΑΣΚΗΣΕΩΝ

Οι ασκήσεις αυτού του φυλλαδίου καλύπτουν τα παρακάτω θέματα και δίνονται ενδεικτικά οι αντίστοιχες ενότητες στο βιβλίο The GNU C Programming Tutorial που μπορείτε συμβουλευτείτε (http://crasseux.com/books/ctutorial/) .

- Δείκτες (κεφάλαιο Pointers)
- Δομές (κεφάλαιο Data Structures)

Άσκηση 1η

Γράψτε μία συνάρτηση *strmix(s,t)* η οποία παίρνει ως παράμετρο τα αλφαριθμητικά *s* και *t* και επιστρέφει ένα νέο αλφαριθμητικό το οποίο περιέχει εναλλάξ τους χαρακτήρες των δύο αλφαριθμητικών. Η συνάρτηση θα πρέπει να δουλεύει με αλφαριθμητικά οποιουδήποτε μήκους (επομένως χρησιμοποιήστε δείκτες). Το κυρίως πρόγραμμα θα ζητάει από το χρήστη δύο αλφαριθμητικά μεγέθους το πολύ N (το N θα το δίνει επίσης ο χρήστης) και θα τυπώνει το αποτέλεσμα της strmix.
Π.χ. (με έντονα η είσοδος του χρήστη):

```
Dwse megisto mege8os string:40
Dwse prwto string (mege8os to poly 40):1234567890
Dwse deytero string (mege8os to poly 40):abcdefg
1a2b3c4d5e6f7g890
```

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

char *strmix(char *s, char *t);

int main()
{
        int N;
        char *s;
        char *t;
        char *mix;

        printf("\nDwse megisto mege8os string:");
        scanf("%d", &N);

        s = (char *) malloc(N*sizeof(char));
        t = (char *) malloc(N*sizeof(char));
        printf("\nDwse prwto string (mege8os to poly %d):", N);
        scanf("%s", s);

        printf("\nDwse deytero string (mege8os to poly %d):", N);
        scanf("%s", t);

        mix=strmix(s,t);

        printf("%s\n",mix);

        free(s);
        free(t);
        free(mix);
}

char *strmix(char *s, char *t)
{
        char *mix;
        int i, slength, tlength, minlength, maxlength;
```

```
        slength = strlen(s);
        tlength = strlen(t);
        maxlength = (slength > tlength) ? slength : tlength;
        minlength = (slength > tlength) ? tlength : slength;

        mix = (char *) malloc((slength+tlength+1)*sizeof(char));

        /* First interpolate chars */
        for (i=0; i<minlength; i++) {
                *(mix+2*i) = *(s+i);
                *(mix+2*i+1) = *(t+i);
        }

        /* Now that the smaller string has no more chars, just add
         the remaining ones from the larger string */
        for (i=0; i<maxlength-minlength; i++) {
                if (maxlength == slength) {
                        *(mix+minlength*2+i) = *(s+minlength+i);
                }
                else {
                        *(mix+minlength*2+i) = *(t+minlength+i);
                }
        }

        return mix;
}
```

Άσκηση 2η

Γράψτε μία συνάρτηση *replace(input, initial, replacement)* η οποία παίρνει ως παράμετρο τα αλφαριθμητικά *input* , *initial* και *replacement* και επιστρέφει ένα νέο αλφαριθμητικό στο οποίο όλες οι εμφανίσεις του *initial* στο *input* έχουν αντικατασταθεί με το *replacement*. Η συνάρτηση θα πρέπει να δουλεύει με αλφαριθμητικά οποιουδήποτε μήκους (επομένως χρησιμοποιήστε δείκτες). Το κυρίως πρόγραμμα θα ζητάει από το χρήστη τρία αλφαριθμητικά μεγέθους το πολύ Ν (το Ν θα το δίνει επίσης ο χρήστης) και θα τυπώνει το αποτέλεσμα της replace.

Π.χ. (με έντονα η είσοδος του χρήστη):

```
Dwse megisto mege8os string:30
Dwse string eisodou (mege8os to poly 30):123456789012345678901234
Dwse string pros antikatastasi(mege8os to poly 30):456
Dwse neo string (mege8os to poly 30):TesseraPenteEksi
123TesseraPenteEksi7890123TesseraPenteEksi78901234
```

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

char *replace(char *input, char *initial, char *replacement);

int main()
{
  int N;
  char *input;
  char *initial;
  char *replacement;
  char *output;

  printf("\nDwse megisto mege8os string:");
  scanf("%d", &N);

  input = (char *) malloc(N*sizeof(char));
  initial = (char *) malloc(N*sizeof(char));
  replacement = (char *) malloc(N*sizeof(char));

  printf("\nDwse string eisodou (mege8os to poly %d):", N);
```

```
  scanf("%s", input);

  printf("\nDwse string pros antikatastasi(mege8os to poly %d):", N);
  scanf("%s", initial);

  printf("\nDwse neo string (mege8os to poly %d):", N);
  scanf("%s", replacement);

  output = replace(input, initial, replacement);

  printf("%s\n",output);

  free(input);
  free(initial);
  free(replacement);
  free(output);
}

char *replace(char *input, char *initial, char *replacement)
{
  int match;
  int i_input; /* Keeps the index position in the input string */
  int i_initial; /* Keeps the index position in the initial string */
  int i_replacement; /* Keeps the index position in the replacement string */
  int i_output; /* Keeps the index position in the output string */
  char *output;

  /* Worst case in terms of output length is that each character in the input
     string is replaced with the replacement string */
  output = (char *) malloc(strlen(input)*strlen(replacement)*sizeof(char));

  i_output = 0;
  for (i_input=0; i_input<strlen(input); i_input++) {

      /* Copy character to the output string */
      *(output+i_output) = *(input+i_input);

      if (*(input+i_input) == *initial) {
          i_initial = 0;
          match = 1;
          /* A first character was matched, let's see for the whole
             initial string */
          while (i_initial<strlen(initial) && match) {
              if (*(input+i_input+i_initial) != *(initial+i_initial)) {
                  match = 0;
              }
              else {
                  i_initial++;
              }
          }

          if (match) {
              /* Now do the replace */
              i_replacement = 0;
              for (i_replacement=0; i_replacement<strlen(replacement); i_replacement++) {
                  *(output+i_output) = *(replacement+i_replacement);
                  i_output++;
              }

              /* Also move the input index to the end of the initial string*/
              i_input += strlen(initial)-1;

              continue;
          }
      }

      /* Copy character to the output string */
      *(output+i_output++) = *(input+i_input);
  }

  return output;
}
```

Άσκηση 3η

Να υλοποιήσετε πρόγραμμα με χρήση δεικτών το οποίο θα επιτρέπει στο χρήστη να επιλέγει μία από τις παρακάτω λογικές πράξεις. Στη συνέχεια ανάλογα με την επιλογή του χρήστη θα ζητά είσοδο η οποία θα δίνεται ως δυαδικός αριθμός. Τέλος θα εφαρμόζει στη δυαδική ακολουθία την επιλεγμένη πράξη και θα εκτυπώνει το αποτέλεσμα.

a. XOR

b. OR

c. AND

d. COMPLEMENT

e Shift left

f. Shift right

Προσοχή, το πρόγραμμα σας δεν θα κάνει χρήση των bitwise operators της C, αλλά θα εφαρμόζει την επιλεγμένη λογική πράξη σε κάθε ψηφίο του δυαδικού αριθμού.

```
#include <stdio.h>
int len (int *number)
{ int *p;
  p=number;
  while (*p!=-1) p++;
  return p-number;
}

void xor(int * op1, int *op2)
{ int len1, len2, i;
  len1=len(op1); len2=len(op2);
  for (i=0; i<len1; i++)
        if (i<=len2) op1[len1-i-1]=(op1[len1-i-1]!=op2[len2-i-1])?1:0;
}

void or(int *op1, int *op2)
{ int len1, len2, i;
  len1=len(op1); len2=len(op2);
  for (i=0; i<len1; i++)
        if (i<=len2) op1[len1-i-1]=((op1[len1-i-1]==1)||(op2[len1-i-1]==1))?1:0;
}

void and(int *op1, int *op2)
{ int len1, len2, i;
  len1=len(op1); len2=len(op2);
  for (i=0; i<len1; i++)
        if (i<=len2) op1[len1-i-1]=((op1[len1-i-1]==0)||(op2[len1-i-1]==0))?0:1;
}

void complement(int *op)
{
   int len1, i;
   len1=len(op);
   for (i=0; i<len1; i++)
          op[i]=(op[i]==1)?0:1;
}

void shiftl(int *op, int k)
{
       int len1, i;
       len1=len(op);
       for (i=0; i<len1-k;i++)
```

```
            op[i]=op[i+k];
      for (i=len1-k; i<len1;i++)
            op[i]=0;
}


void shiftr(int *op, int k)
{
      int len1, i;
      len1=len(op);
      for (i=0; i<len1-k;i++)
            op[len1-1-i]=op[len1-1-i-k];
      for (i=0; i<k;i++)
            op[i]=0;
}

void convertToArray(int* array, char* s, int N)
{
 int j,i;
 int l = strlen(s);

 for(i = l-1; i >=0; i--)
 {
    if (s[l-1-i] == '1')
       array[i] = 1;
    else
       array[i] = 0;
 }

 for(j = N-1; j >= strlen(s); j--)
    array[j] = -1;


}

void printBinary(int *a, int n)
{
  int i;

  for (i=n-1; i >= 0; i--)
    if (a[i] != -1)
       printf("%d", a[i]);
  printf("\n");
}

void getString(char *s)
{
   int i = 0;
   do{
        s[i++] = getchar();
   } while(s[i-1] != '\n');
   s[i-1] = '\0';
}

main()
{
    char *n1, *n2;
    int *A, *B;
    int shift, N, option;

    printf("Dwse megisto arithmo bits: ");
    scanf("%d", &N);

    n1 = (char *)malloc(N*sizeof(char));
    n2 = (char *)malloc(N*sizeof(char));
    A = (int *)malloc(N*sizeof(int));
    B = (int *)malloc(N*sizeof(int));
```

```
    do
    {
        printf("1. XOR\n");
        printf("2. OR\n");
        printf("3. AND\n");
        printf("4. COMPLEMENT\n");
        printf("5. SHIFT LEFT\n");
        printf("6. SHIFT RIGHT\n");
        printf("0. Exodos\n");
        printf("Dwse epilogh:");
        scanf("%d", &option);
        fflush(stdin);
            if (option >=1 && option <= 3)
            {
               printf("Dwse 1o dyadiko (mexri %d bits):", N);
               getString(n1);

               printf("Dwse 2o dyadiko (mexri %d bits):", N);
               getString(n2);

               convertToArray(A, n1, N);
               convertToArray(B, n2, N);

               if (option == 1)
               {
                  xor(A, B);
                  printf("Apotelesma XOR: ");
                  printBinary(A, N);
               }
               else if (option == 2)
               {
                  or(A, B);
                  printf("Apotelesma OR: ");
                  printBinary(A, N);
               }
               else
               {
                  and(A, B);
                  printf("Apotelesma AND: ");
                  printBinary(A, N);
               }
            }
            else if (option>=4 && option <=6)
            {
                printf("Dwse dyadiko (mexri %d bits):", N);
                getString(n1);
                convertToArray(A, n1, N);

                if(option == 5 || option == 6)
                {
                    printf("Dwse theseis shift: ");
                    scanf("%d", &shift);
                }

                if(option == 4)
                {
                  complement(A);
                  printf("Apotelesma COMPLEMENT: ");
                  printBinary(A, N);
                }
                else if (option == 5)
                {
                  shiftl(A, shift);
                  printf("Apotelesma SHIFT LEFT: ");
                  printBinary(A, N);
                }
                else if (option == 6)
                {
```

```
                    shiftr(A, shift);
                    printf("Apotelesma SHIFT RIGHT: ");
                    printBinary(A, N);
                }
            }

    }while(option != 0);

}
```

Άσκηση 4η

Σας ζητείτε να αποθηκεύσετε σε έναν πίνακα μεγέθους N στοιχεία τύπου struct employee. Κάθε καινούριο στοιχείο αποθηκεύεται στην τελευταία άδεια θέση του πίνακα και μπορεί να διαγραφεί μόνο το τελευταίο στοιχείο που έχει εισαχθεί στον πίνακα. Δίνεται:

```
struct employee
{
        char *firstName;
        char *lastName;
        int age;
};
```

Υλοποιήστε τις παρακάτω συναρτήσεις:

```
/* δημιουργία του πίνακα*/
/* pos : πρώτη ελεύθερη θέση στον πίνακα */
/* επιστρέφει πίνακα με N στοιχεία τύπου struct employee* */
struct employee* create( int *pos);

/* εισαγωγή στοιχείου item στον πίνακα pin */
/* pos : πρώτη ελεύθερη θέση στον πίνακα */
void ins(struct employee* pin, int *pos, struct employee* item);

/* διαγραφή στοιχείου από τον πίνακα */
/* pos : πρώτη ελεύθερη θέση στη στοίβα */
struct employee* del(struct employee* pin, int *pos);

/*εκτύπωση πίνακα */
/* pos : πρώτη ελεύθερη θέση στη στοίβα */
void printpin(struct employee* pin, int *pos);

/* ταξινόμηση πίνακα */
/* pos : πρώτη ελεύθερη θέση στη στοίβα */
/* field : το πεδίο με βάση το οποίο θέλουμε να γίνει η ταξινόμηση */
/* direction : αν η ταξινόμηση θα είναι αύξουσα τότε (direction=1) */
/* Αν θα είναι φθίνουσα (direction=-1) αντίστοιχα. */
void sort(struct employee* pin, int *pos, char *field, int direction);
```

Γράψτε την main() η οποία εμφανίζει μενού επιλογών στο χρήστη:

1. Εισαγωγή στοιχείων υπαλλήλου
2. Διαγραφή στοιχείων υπαλλήλου
3. Εμφάνιση όλων των υπαλλήλων
4. Ταξινόμηση Πίνακα
5. Έξοδος

Ανάλογα με την επιλογή του χρήστη ζητούνται τα απαραίτητα στοιχεία (π.χ. για την Εισαγωγή Στοιχείων ζητούνται από τον χρήστη το όνομα, επώνυμο και ηλικία) και καλείται η κατάλληλη συνάρτηση.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define N 100

typedef struct employee employee;

struct employee
{
    char *firstName;
    char *lastName;
    int age;
};

struct employee* create( int *pos)
{
    struct employee *head = (struct employee*) malloc(N*sizeof(employee) );
    *pos = 0;
    return head;
}

void ins( struct employee *pin, int *pos, struct employee *item)
{
    if ( pin != NULL )
        if ( (*pos) < N)
        {
            pin[(*pos)].firstName = item->firstName;
            pin[(*pos)].lastName = item->lastName;
            pin[(*pos)].age = item->age;
            (*pos)++;
        }
        else
            printf("pin full\n");
        else
            printf("pin not initialized\n");
}

struct employee* del( struct employee *pin, int *pos)
{
    if ( pin != NULL )
        if ( *pos != 0)
        {
            (*pos)--;
            return &pin[*pos];
        }
        else
        {
            printf("pin empty\n");
            return NULL;
        }
        else
        {
            printf("pin not initialized\n");
            return NULL;
        }
}

void printpin( struct employee *pin, int *pos)
```

```
{
    int i = 0;

    if ( pin != NULL )
        for ( i = 0; i < *pos; i++)
        {
            printf("%3d. ", i+1);
            printf("%s ", pin[i].firstName);
            printf("%s ", pin[i].lastName);
            printf("%d\n", pin[i].age);
        }
        else
            printf( "pin not initialized\n");
}


void swap( struct employee *pin, int *pos, int first, int second)
{
    struct employee tmp;

    tmp.firstName = strdup( pin[first].firstName );
    tmp.lastName = strdup( pin[first].lastName );
    tmp.age = pin[first].age;

    pin[first].firstName = strdup( pin[second].firstName );
    pin[first].lastName = strdup( pin[second].lastName );
    pin[first].age = pin[second].age;

    pin[second].firstName = strdup( tmp.firstName );
    pin[second].lastName = strdup( tmp.lastName );
    pin[second].age = tmp.age;
}

void sortStrFirst( struct employee *pin, int *pos, int direction)
{
    int i = 0, j = 0, tmp = 0;

    for( j = 0; j < *pos-1; j++)
        for( i = j; i < *pos-1; i++)
        {
            tmp = j;
            if ( direction == 1)
            {
                if ( strcmp( pin[tmp].firstName, pin[i+1].firstName) > 0 )
                {
                    swap( pin, pos, tmp, i+1);
                    tmp = i+1;
                }
            }
            else
            {
                if ( strcmp( pin[tmp].firstName, pin[i+1].firstName) < 0 )
                {
                    swap( pin, pos, tmp, i+1);
                    tmp = i+1;
                }
            }
        }
}

void sortStrLast( struct employee *pin, int *pos, int direction)
{
    int i = 0, j = 0, tmp = 0;

    for( j = 0; j < *pos-1; j++)
        for( i = j; i < *pos-1; i++)
        {
            tmp = j;
            if ( direction == 1)
            {
                if ( strcmp( pin[tmp].lastName, pin[i+1].lastName) > 0 )
                {
                    swap( pin, pos, tmp, i+1);
                    tmp = i+1;
                }
            }
            else
```

```
            {
                if ( strcmp( pin[tmp].lastName, pin[i+1].lastName) < 0 )
                {
                    swap( pin, pos, tmp, i+1);
                    tmp = i+1;
                }
            }
        }
}

void sortIntAge( struct employee *pin, int *pos, int direction)
{
    int i = 0, j = 0, tmp = 0;

    for( j = 0; j < *pos-1; j++)
        for( i = j; i < *pos-1; i++)
        {
            tmp = j;
            if ( direction == 1)
            {
                if ( pin[tmp].age > pin[i+1].age )
                {
                    swap( pin, pos, tmp, i+1);
                    tmp = i+1;
                }
            }
            else
            {
                if ( pin[tmp].age < pin[i+1].age )
                {
                    swap( pin, pos, tmp, i+1);
                    tmp = i+1;
                }
            }
        }
}

int cmp_firstname(const void *a, const void *b)
{
  struct employee *ia  = (struct employee *)a;
  struct employee *ib  = (struct employee *)b;
  printf("%d\n", strcmp((*ia).firstName, (*ib).firstName));
 return strcmp((*ia).firstName, (*ib).firstName);
/* strcmp functions works exactly as expected from
comparison function */
}

int cmp_lastname(const void *a, const void *b)
{
 struct employee *ia = (struct employee *)a;
 struct employee *ib = (struct employee *)b;
 return strcmp((*ia).lastName, (*ib).lastName);
/* strcmp functions works exactly as expected from
comparison function */
}

int cmp_age(const void *a, const void *b)
{
 struct employee *ia = (struct employee *)a;
 struct employee *ib = (struct employee *)b;
 return ((*ia).age - (*ib).age);
}

int cmp_firstname_desc(const void *a, const void *b)
{
  struct employee *ia  = (struct employee *)a;
  struct employee *ib  = (struct employee *)b;
 return -1*(strcmp((*ia).firstName, (*ib).firstName));
/* strcmp functions works exactly as expected from
comparison function */
}

int cmp_lastname_desc(const void *a, const void *b)
{
 struct employee *ia = (struct employee *)a;
 struct employee *ib = (struct employee *)b;
 return -1*(strcmp((*ia).lastName, (*ib).lastName));
```

```
/* strcmp functions works exactly as expected from
comparison function */
}

int cmp_age_desc(const void *a, const void *b)
{
 struct employee *ia = (struct employee *)a;
 struct employee *ib = (struct employee *)b;
 return ((*ib).age - (*ia).age);
/* strcmp functions works exactly as expected from
comparison function */
}

void sort(struct employee* pin, int *pos, char *field, int direction)
{
   /* if ( !strcmp( field, "firstName") )
        sortStrFirst( pin, pos, direction);
    if ( !strcmp( field, "lastName") )
       sortStrLast( pin, pos, direction);
    if ( !strcmp( field, "age") )
        sortIntAge( pin, pos, direction);
   */
    /* Use qsort of stdlib */


    if ( !strcmp( field, "firstName") )
      if (direction == 1)
        qsort(pin, (size_t)(*pos), sizeof(struct employee), cmp_firstname);
      else
        qsort(pin, (size_t)(*pos), sizeof(struct employee), cmp_firstname_desc);
    if ( !strcmp( field, "lastName") )
       if (direction == 1)
        qsort(pin, (size_t)(*pos), sizeof(struct employee), cmp_lastname);
      else
        qsort(pin, (size_t)(*pos), sizeof(struct employee), cmp_lastname_desc);
    if ( !strcmp( field, "age") )
       if (direction == 1)
        qsort(pin, (size_t)(*pos), sizeof(struct employee), cmp_age);
      else
        qsort(pin, (size_t)(*pos), sizeof(struct employee), cmp_age_desc);

}

int main()
{
    struct employee *head;
    int pos, f, o, command, i;
    struct employee *tmp;
    struct employee a;



    head = create( &pos);

    do
    {
        printf("\n\n Ari8mos stoixeiwn ston pinaka %d \n\n", pos);
        printf("1. Eisagwgh Stoixeiwn Ypallhlou\n");
        printf("2. Diagrafh Stoixeiwn Ypallhlou\n");
        printf("3. Emfanish olwn twn Ypallhlwn\n");
        printf("4. Taxinomisi\n");
        printf("5. Exit\n");
        do{
            printf("Dwse epilogh: ");
            scanf("%d", &command);
        }while(!(command>0 && command < 6));

        if (command == 1)
        {
             a.firstName = (char *)malloc(255*sizeof(char));
             a.lastName = (char *)malloc(255*sizeof(char));
             printf("\n\n Dwse Stoixeia Ypallhlou: \n\n");
             printf(" Onoma: ");
             scanf("%s", a.firstName);
             printf(" Epwnymo: ");
             scanf("%s", a.lastName);
             printf(" Hlikia: ");
```

```
            scanf("%d", &a.age);
            ins( head, &pos, &a);
        }
        else if (command == 2)
        {
            printf("\n Diagrafh Teleutaiou sth Lista Ypallhlou: ");
            tmp = del(head, &pos);
        }
        else if (command == 3)
        {
            printf("Emfanish olwn twn ypallhlwn: ");
            printpin( head, &pos);
        }
        else if (command == 4)
        {
            do{
                printf("1. Taxinomisi me onoma\n");
                printf("2. Taxinomisi me eponimo\n");
                printf("3. Taxinomisi me hlikia\n");
                printf("Dwse epilogh: \n");
                scanf("%d", &f);
            }while(!(f > 0 && f < 4));

            do
            {
                printf(" 1. Taxinomisi Auxousa\n");
                printf("-1. Taxinomisi Fthinousa\n");
                printf("Dwse epilogh: \n");
                scanf("%d", &o);
            }while(!(o == 1 || o == -1));

            switch(f){
               case 1: sort( head, &pos, "firstName", o);
                       break;
               case 2: sort( head, &pos, "lastName", o);
                       break;
               case 3: sort( head, &pos, "age", o);
                       break;
            }

            printpin( head, &pos);
        }

    }while (command != 5);

    return 0;
}
```

Άσκηση 5η

Δημιουργείστε κώδικα ο οποίος υλοποιεί το ακόλουθο παιχνίδι: αρχικά ζητείται από τον πρώτο χρήστη να εισαγάγει μια λέξη. Στη συνέχεια η οθόνη καθαρίζεται και εμφανίζονται χαρακτήρες '_' στη θέση των γραμμάτων της λέξης καθώς και ένας μετρητής που δηλώνει πόσες προσπάθειες απομένουν στο δεύτερο χρήστη, που καλείται να μαντέψει τη λέξη δίνοντας έναν χαρακτήρα κάθε φορά. Αν ο εισαγόμενος χαρακτήρας υπάρχει εμφανίζεται στην κατάλληλη θέση της λέξης, αν δεν υπάρχει ο μετρητής μειώνεται κατά 1 και αν έχει επιλεχθεί ήδη εμφανίζεται ένα ενημερωτικό μήνυμα. Το παιχνίδι τελειώνει όταν βρεθούν όλα τα γράμματα της λέξης ή όταν μηδενιστεί ο μετρητής (οπότε εμφανίζονται τα κατάλληλα μηνύματα στην οθόνη).
Προτείνεται:

1. να χρησιμοποιηθεί κώδικας για τη μετατροπή πεζών σε κεφαλαία σε μορφή συνάρτησης έτσι ώστε όλοι οι εισαγόμενοι χαρακτήρες να μετατρέπονται σε κεφαλαία

2. ο έλεγχος ύπαρξης ενός χαρακτήρα σε μια λέξη να γίνεται με την κλήση συνάρτησης η οποία θα λαμβάνει ως ορίσματα τη λέξη και τον χαρακτήρα προς αναζήτηση και θα επιστρέφει 0 αν δεν υπάρχει ή 1 αν υπάρχει.

Υπόδειξη: Ο καθαρισμός της οθόνης γίνεται με την εντολή system("clear") (unix) ή system("cls") (DOS)

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

int HandleGivenLetter(char KeyLetter, char *HidWord, char *RevWord)
{
	int i, lcount = 0;
	for (i = 0; i < strlen(HidWord); i++)
	{
		// an exei hdh bre8ei to gramma (epeidh to proteine se
		// prohgoumeno xrono o xrhsths) epistrefoume 0 emfaniseis
		if(KeyLetter == RevWord[i])
			return 0;
		else if(KeyLetter == HidWord[i])
		{
			// an bre8ei to gramma, to emfanizoume sth RevWord
			// kai auksanoume to ari8mo twn emfanisewn tou
			RevWord[i] = KeyLetter;
			lcount++;
		}
	}
	return lcount;
}

int main()
{
	char hiddenWord[50], revealedWord[50], givenLetter;
	int i, HiddenChars, userpoints, LetterCount;

	userpoints = 10;

	printf("Xrhsth 1::Dwse thn Krymmenh Leksh: ");
	// apo8hkeuoume thn krymmenh leksh
	scanf("%s", hiddenWord);

	HiddenChars = strlen(hiddenWord);

	// bazoume '-' sthn leksh pou 8a deixnoume sto xrhsth 2
	for(i=0; i < HiddenChars; i++){
		revealedWord[i] = '-';
	}

	revealedWord[HiddenChars] = 0;
	system("clear");

	while( userpoints > 0 && HiddenChars > 0)
	{
		printf("%s\t\t Points Left %d\nXrhsth 2::Proteine kapoio gramma: \n",
			revealedWord, userpoints);
		scanf("\n%c", &givenLetter);
		LetterCount = HandleGivenLetter(givenLetter, hiddenWord, revealedWord);

		if(LetterCount)
		{
			printf("Bre8hkan %d '%c' sth leksh.\n", LetterCount,
				givenLetter);

			// meiwnoume ton ari8mo twn krymmenw xarakthrwn
			HiddenChars -= LetterCount;
```

```
            }
            else
            {
                printf("Den bre8hkan '%c' sth leksh.\n", givenLetter);
                // apotyxia shmainei meiwsh twn pontwn
                userpoints--;
            }
      }

      if(!HiddenChars) printf("Bravo! ");
      else printf("Apotyxia! ");

     // kai emfanish ths krymmenhs lekshs
     printf("H krymmenh leksh htan: \"%s\" \n", hiddenWord);

     return 0;
}
```