Δομές Δεδομένων Εργαστήριο

Λύσεις Ασκήσεων Εργαστηριακού Οδηγού Β Έκδοση 2018

Ηράκλειο 2021

ΕΡΓΑΣΤΗΡΙΟ 1

Α'. ΠΕΡΙΕΧΟΜΕΝΑ ΜΑΘΗΜΑΤΟΣ

Στο εργαστήριο 1 καλύπτονται τα παρακάτω θέματα:

- **Πίνακες** μιας διάστασης.
- Μονοδιάστατοι **πίνακες** ως **ορίσματα συναρτήσεων.**

Β'. ΑΣΚΗΣΕΙΣ ΓΙΑ ΕΚΤΕΛΕΣΗ – ΕΠΙΔΕΙΞΗ / ΕΠΕΞΗΓΗΣΕΙΣ

5. Να γραφεί πρόγραμμα, στο οποίο να αθροίζονται μεταξύ τους τα στοιχεία δύο πινάκων. Να δημιουργηθούν τρείς συναρτήσεις. Η πρώτη input() χρησιμοποιείται για να βάλουμε τιμές στους δύο πίνακες, η δεύτερη addarray() για να γίνει η πρόσθεση και η τρίτη display() για να εμφανίσουμε στην οθόνη τον τρίτο πίνακα με το αποτέλεσμα της πρόσθεσης.

```c
#include <stdio.h>
#define  N  10

void input(int*);
void display(int*);
void addarray(int*, int*, int*);

int main () {
    int a[N], b[N], c[N];
    printf ("Eisagwgi stoixeiwn 1ou Pinaka \n");
    input(a);
    printf ("Eisagwgi stoixeiwn 2ou Pinaka \n");
    input(b);
    addarray(a, b, c);
    printf ("To athroisma twn Stoixeiwn twn Pinakwn einai: \n");
    display(c);
    return 0;
}

void input (int c[] ) {
    int i;
    for (i = 0; i < N; i++) {
```

```
        scanf("%d", &c[i]);
    }
}

void addarray (int a[], int b[], int c[]) {
    int  i;
    for (i = 0; i < N; i++) {
        c[i] = a[i] + b[i];
    }
}

void display (int d[]) {
    int i;
    for (i = 0; i < N; i++) {
        printf(" %d\n ", d[i]);
    }
}
```

6. Να γραφεί πρόγραμμα, στο οποίο να διαβάζονται float από το πληκτρολόγιο και καταχωρούνται σε ένα πίνακα, τον c. Οι float αυτοί υποτίθεται ότι αντιπροσωπεύουν ηλικίες κάποιων ατόμων. Στη συνέχεια, θα καλείται μια συνάρτηση, η avg( ), η οποία θα βρίσκει τον μέσο όρο των ηλικιών και θα τον επιστρέφει στην main( ), από όπου και θα εμφανίζεται στην οθόνη.

```
#include <stdio.h>
#define N 10

float average (float []);

int main () {
    float  avg, c[N];
    int  k;

    for (k = 0; k < N; k++) {
        scanf ("%f", &c[k]);
    }
    avg = average(c);
    printf ("Average age=%.2f", avg);

    return 0;
}


float average (float a[]) {
    int i;
    float avg, sum = 0.0;
```

```
    for (i = 0; i < N; i++) {
        sum += a[i];
    }
    avg = sum / N;
    return avg;
}
```

7. Έστω ένας πίνακας ακεραίων, ο list, με N θέσεις. Ο πίνακας να πάρει τιμές στη main( ) από το πληκτρολόγιο. Να καλείται μια συνάρτηση, η sort( ), η οποία θα ταξινομήσει τα στοιχεία του πίνακα. Ο ταξινομημένος πίνακας να εμφανίζεται στην οθόνη από την main( ).

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 10

void sort (int*, int);

int main () {
    int list[SIZE];
    int k, meg, ak,i;
    for (k = 0; k < SIZE; k++) {
        scanf("%d", &ak);
        if (ak == 0) {
            break;
        }
        else {
            list[k] = ak;
        }
    }
    sort(list, k);

    for (k = 0; k < SIZE; k++) {
        printf("%d\n", list[k]);
    }
    return 0;
}

void sort (int slist[], int snum) {
    int sout, sin, stemp;

    for (sout = 0; sout < snum - 1; sout++) {
        for (sin = sout + 1; sin < snum; sin++) {
            if (slist[sout] > slist[sin]) {
                stemp = slist[sin];
```

```
                slist[sin] = slist[sout];
                slist[sout] = stemp;
            }
        }
    }
}
```

ΕΡΓΑΣΤΗΡΙΟ 2

Α'. ΠΕΡΙΕΧΟΜΕΝΑ ΜΑΘΗΜΑΤΟΣ

Στο εργαστήριο 2 καλύπτονται τα παρακάτω θέματα:

- Πίνακες δύο διαστάσεων.
- Πίνακες συμβολοσειρών.
- Συμμετρικοί, τριγωνικοί πίνακες.
- Αραιοί πίνακες.

Β'. ΑΣΚΗΣΕΙΣ ΓΙΑ ΕΚΤΕΛΕΣΗ – ΕΠΙΔΕΙΞΗ / ΕΠΕΞΗΓΗΣΕΙΣ

1. Να γραφεί ένα πρόγραμμα, στο οποίο να δηλώσετε ένα δισδιάστατο πίνακα ακεραίων, τον **grades[ ][ ].** Ο πίνακας να αποτελείται το πολύ από τόσες γραμμές, όσοι είναι οι μαθητές μιας τάξης και από 5 στήλες, όσος είναι ο αριθμός των τεστ, στα οποία έχουν υποβληθεί οι μαθητές.
Να δώσετε τιμές στον πίνακα grades και στη συνέχεια να δημιουργήσετε μια συνάρτηση, την **better( )**, η οποία να βρίσκει και να επιστρέφει στη main( ) τον μεγαλύτερο από τους βαθμούς αυτούς, ο οποίος και να εμφανίζεται στην οθόνη. Ο πίνακας grades δεν είναι απαραίτητο να γεμίσει ολόκληρος ως προς τον αριθμό των μαθητών. Όταν καλείται η συνάρτηση better( ), αυτή ενημερώνεται για το πόσες γραμμές του πίνακα να ερευνήσει (δηλαδή για το πόσοι μαθητές εξετάστηκαν στα τεστ).

```c
/* A' Τρόπος */
#include <stdio.h>
#define STUDENTS 15
#define TESTS 5
int better (int[][TESTS], int);

int main () {
    int grades[STUDENTS][TESTS], highest, i, j;
    int num_students = 5;
    int num_tests = 4;

    for (i = 0; i < num_students; i++) {
        for (j = 0; j < num_tests; j++) {
```

```
            scanf("%d", &grades[i][j]);
        }
    }
    highest = better(grades, num_students, num_tests);
    printf("O upsiloteros vathmos einai %d.\n", highest);
    return 0;
}

int better (int a[][TESTS], int row, int col) {
    int i, j;
    int highest = a[0][0];
    for (i = 0; i < row; i++)
        for (j = 0; j < col; j++)
            if (a[i][j] > highest)
                highest = a[i][j];
    return highest;
}

/* B' Τρόπος */
#include <stdio.h>
#define STUDENTS 15
#define TESTS 5

int better (int*, int);

int main () {
    int grades[STUDENTS][TESTS];
    int highest, i, j;
    int num_students = 5;
    int num_tests = 4;
    for (i = 0; i < num_students; i++) {
        for (j = 0; j < num_tests; j++) {
            scanf("%d", &grades[i][j]);
        }
    }
    highest = better(grades[0], num_students * num_tests);
    printf("O upsiloteros vathmos einai %d.\n", highest);
    return 0;
}

int better (int *a, int elem) {
    int i, j;
    int highest = *a;

    for (i = 0; i < elem; i++) {
        if (*(a + i) > highest) {
            highest = *(a + i);
        }
```

```
    }
    return highest;
}
```

Γ'. ΣΥΜΠΛΗΡΩΜΑΤΙΚΕΣ ΑΣΚΗΣΕΙΣ ΓΙΑ ΕΞΑΣΚΗΣΗ:

4. Στη main( ) ενός προγράμματος να δηλώσετε ένα δισδιάστατο πίνακα χαρακτήρων, RxC, τον pin. Στον πίνακα αυτόν να καταχωρείτε συμβολοσειρές. Θα καταχωρήσετε R το πολύ συμβολοσειρές ή αλλιώς θα σταματάτε όταν δώσετε κενή συμβολοσειρά. Να εμφανίσετε στην οθόνη το σύνολο των χαρακτήρων που δόθηκαν, καθώς και το πλήθος των χαρακτήρων της μεγαλύτερης συμβολοσειράς.

```
#include<stdio.h>
#include<string.h>
#include <stdlib.h>
#define R 5
#define C 50

int main () {
    char pinax[R][C];
    int i, j, k, count = 0, max = 0, cnt = 0;

    for (i = 0; i < R; i++) {
        printf ("Dwste ti %dh leksi\n", i + 1);
        gets(pinax[i]);
        if (strlen(pinax[i]) == 0) {
            break;
        }
    }

    for (k = 0; k < i; k++) {
        for (j = 0; pinax[k][j]!='\x0'; j++) {
            count++;
            cnt++;
        }
        if (cnt > max) {
            max = cnt;
        }
        cnt = 0;
    }
    printf ("To sunolo twn xaraktirwn pou dwsate einai %d\n", count)
;
    printf ("H megaluteri sumboloseira exei %d xaraktires\n", max);
```

```
    return 1;
}
```

ΕΡΓΑΣΤΗΡΙΟ 3

Α’. ΠΕΡΙΕΧΟΜΕΝΑ ΜΑΘΗΜΑΤΟΣ

Στο εργαστήριο 3 καλύπτονται τα παρακάτω θέματα:

- Δομές (structures): Περιγραφή, πεδία δομής, δηλώσεις και δεδομένα
- Φωλιασμένες δομές.
- Πίνακες δομών.
- Δομές ως παράμετροι και ως τιμή επιστροφής συναρτήσεων.
- Δείκτες σε δομές.

Β’. ΑΣΚΗΣΕΙΣ ΓΙΑ ΕΚΤΕΛΕΣΗ – ΕΠΙΔΕΙΞΗ / ΕΠΕΞΗΓΗΣΕΙΣ

Δομές ως παράμετροι και ως τιμή επιστροφής συναρτήσεων.

4. Σε ένα πρόγραμμα έχετε τον εξής τύπο δομών:

```
struct stype {
    int j;
    char ch[30];
    float fp; };
```

Να δηλώσετε στη main( ) δυο μεταβλητές δομές του πιο πάνω τύπου, τις str1 και str2 και να τους δώσετε τιμές από το πληκτρολόγιο. Στα πεδία ch να καταχωρήσετε συμβολοσειρές. Στη συνέχεια να καλέσετε μια συνάρτηση, η οποία θα λέγεται struct_swap( ). Η συνάρτηση θα καλείται μια μόνο φορά στη main( ) και θα εναλλάσσει τα περιεχόμενα των str1 και str2. Η συνάρτηση να δηλωθεί, να οριστεί.

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

struct stype {
    int   j;
    char  ch[30];
    float fp; };

void struct_swap(struct stype*, struct stype*);

int main() {
```

```
    struct stype str1, str2;
    char pin[10];

    printf("Gemisma domwn\n");
    printf("Akeraios domhs str1 ");
    gets(pin);
    str1.j = atoi(pin);
    printf("Symboloseira domhs str1 ");
    gets(str1.ch);
    printf("Float domhs stri ");
    gets(pin);
    str1.fp = atof(pin);
    printf("Akeraios domhs str2 ");
    gets(pin);
    str2.j = atoi(pin);
    printf("Symboloseira domhs str2 ");
    gets(str2.ch);
    printf("Float domhs str2 ");
    gets(pin);
    str2.fp = atof(pin);
    printf("*****************************\n");
    struct_swap(&str1, &str2);
    printf("Akeraios domhs str1 %d\n", str1.j);
    printf("Symboloseira domhs str1 %s\n", str1.ch);
    printf("Float domhs str1 %f\n", str1.fp);
    printf("Akeraios domhs str2 %d\n", str2.j);
    printf("Symboloseira domhs str2 %s\n", str2.ch);
    printf("Float domhs str2 %f\n", str2.fp);
    return 1;
}

void struct_swap(struct stype *s1, struct stype *s2) {
    struct stype temp;
    temp = *s1;
    *s1 = *s2;
    *s2 = temp;
}
```

5. Σε ένα πρόγραμμα έχετε τον εξής τύπο δομών:

```
struct funds {
    char name[20];
    float poson; };
```

Να δηλώσετε στη main( ) ένα πίνακα Ν δομών του πιο πάνω τύπου, τον persons και να του δώσετε τιμές από το πληκτρολόγιο. Στο πεδίο name να καταχωρήσετε συμβολοσειρές. Στη συνέχεια:

- Να διαβάσετε ένα χαρακτήρα στη main( ), τον ch.
- Να καλέσετε μια συνάρτηση, την athroisma( ), η οποία θα καλείται μια μόνο φορά στη main( ) και θα κάνει τα εξής:
  - Θα ελέγχει ποιες από τις συμβολοσειρές των στοιχείων του πίνακα δομών έχουν πρώτο γράμμα τον χαρακτήρα ch.
  - Θα δημιουργεί το άθροισμα των πεδίων poson αυτών των στοιχείων του πίνακα.
  - Θα επιστρέφει το άθροισμα στη main( ).

Η main( ) να εμφανίζει στην οθόνη την τιμή που επέστρεψε η athroisma( ).

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#define N 5

struct funds  {
    char  name[20];
    float poson; };

float athroisma(struct funds *, char);

int main() {
    struct funds persons[N];
    int k;
    float sum;
    char ch, temp[10];

    printf("Gemisma pinaka domwn\n");
    for (k = 0; k < N; k++)  {
        printf("Symboloseira domhs %d: ", k + 1);
        gets(persons[k].name);
        printf("Float domhs %d: ", k + 1);
        gets(temp);
        persons[k].poson = atof(temp);
    }
    printf("Dwste xarakthra\n");
    ch = getche();
    sum = athroisma(persons, ch);
    printf("%5.2f\n", sum);
    return 1;
}

float athroisma(struct funds *apersons, char ach) {
    float asum = 0;
```

```
    int k;

    for (k = 0; k < N; k++) {
        if ((apersons + k)->name[0] == ach)
            asum += (apersons + k)->poson;
    }
    return asum;
}
```

ΕΡΓΑΣΤΗΡΙΟ 4

Α’. ΠΕΡΙΕΧΟΜΕΝΑ ΜΑΘΗΜΑΤΟΣ

Στο εργαστήριο 3 καλύπτονται τα παρακάτω θέματα:

- Δυναμική δέσμευση μνήμης.
- Συναρτήσεις malloc, calloc, realloc.
- Πίνακες δεικτών.
- Δυναμικοί πίνακες.

Β’. ΑΣΚΗΣΕΙΣ ΓΙΑ ΕΚΤΕΛΕΣΗ – ΕΠΙΔΕΙΞΗ / ΕΠΕΞΗΓΗΣΕΙΣ

2. Να διαβάσετε μια συμβολοσειρά από το πληκτρολόγιο, την pin. Να δεσμεύσετε με την malloc( ) όσο χώρο χρειάζεστε για την αποθήκευση της συμβολοσειράς αυτής. Στη συνέχεια να διαβάσετε μια άλλη συμβολοσειρά, την mat, την οποία να αποθηκεύσετε στον χώρο της pin, μεγαλώνοντάς τον ή μικραίνοντάς τον με την χρήση της realloc( ). Στη συνέχεια να γράψετε στην οθόνη την pin.

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

int main() {
    char pin[80], mat[80], *mes;
    // Read string 1
    puts("Dwste prwth symboloseira");
    gets(pin);
    // Allocate and copy string 1
    mes = (char *) malloc(strlen(pin) + 1);
    strcpy(mes, pin);
    // Read string 2
    puts("Dwste deyterh symboloseira");
    gets(mat);
    // Reallocate and copy string 2
    mes = (char *) realloc(mes, (strlen(mat) + 1));
    strcpy (mes,mat);
    puts(mes);
    return 1;
}
```

3. Να γράψετε ένα πρόγραμμα, στο οποίο θα κάνετε τα εξής:

Να δηλώσετε ένα πίνακα δεικτών σε χαρακτήρα MAX θέσεων. Στη συνέχεια να διαβάζετε συμβολοσειρές, για κάθε μια από τις οποίες να δεσμεύετε όσο χώρο χρειάζεται (με την malloc ή την calloc) και να τοποθετήσετε στην κάθε μία έναν από τους δείκτες του πίνακα δεικτών. Συμβολοσειρές να διαβάζετε μέχρι να διαβάσετε MAX συνολικά ή μέχρι να δώσετε κενή συμβολοσειρά.

Αφού τελειώσετε το διάβασμα, να εμφανισθούν στην οθόνη όλες οι συμβολοσειρές που διαβάσατε.

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

#define MAX 100
#define LINE 80

int main() {
    char temp[LINE];
    char *names[MAX];
    int k = 0, j;

    puts("DWSTE ONOMA: ");
    gets(temp);

    while (temp[0] != '\x0') {
        names[k] = (char *) malloc(strlen(temp) + 1);
        strcpy(names[k], temp);
        k++;
        if (k == MAX) {
            puts ("O PINAKAS GEMISE");
            break;
        }
        else {
            puts("DWSTE EPOMENO ONOMA");
            gets(temp);
        }
    }
    putchar('\n');
    puts("KATALOGOS ONOMATWN");
    for (j = 0; j < k; j++)
        puts(names[j]);
    return 1;
}
```

ΕΡΓΑΣΤΗΡΙΟ 5

Α'. ΠΕΡΙΕΧΟΜΕΝΑ ΜΑΘΗΜΑΤΟΣ

Στο εργαστήριο 5 καλύπτονται τα παρακάτω θέματα:

- Στοίβες**.** Υλοποίηση με πίνακα.
- Ουρές αναμονής και ουρές προτεραιότητας**.** Υλοποίηση με πίνακα.

Β'. ΑΣΚΗΣΕΙΣ ΓΙΑ ΕΚΤΕΛΕΣΗ – ΕΠΙΔΕΙΞΗ / ΕΠΕΞΗΓΗΣΕΙΣ

Στοίβες:

1. Να γραφεί ένα πρόγραμμα, στο οποίο να χρησιμοποιείτε μια στοίβα θετικών ακεραίων, την οποία να υλοποιήσετε με την χρήση πίνακα ακεραίων Ν θέσεων. Να γράψετε τις συναρτήσεις ώθησης (push) και ανάκλησης (pop) από την στοίβα, στις οποίες να γίνεται έλεγχος πλήρους και κενής στοίβας αντίστοιχα. Το πρόγραμμα να διαβάζει ένα χαρακτήρα από το πληκτρολόγιο, τον ch και:

- Εάν ο χαρακτήρας είναι το U να διαβάζετε ένα ακέραιο και να γίνεται ώθηση του ακεραίου στην στοίβα.
- Εάν ο χαρακτήρας είναι το Ο να γίνεται ανάκληση ακεραίου από την στοίβα, ο οποίος να εμφανίζεται στην οθόνη.
- Εάν ο χαρακτήρας είναι το Ε, να τελειώνει η όλη διαδικασία και να εμφανίζονται στην οθόνη τα περιεχόμενα της στοίβας.
- Εάν ο χαρακτήρας δεν είναι κανείς από τους παραπάνω, να γράφεται στην οθόνη «ΛΑΘΟΣ ΧΑΡΑΚΤΗΡΑΣ» και να διαβάζεται νέος χαρακτήρας από το πληκτρολόγιο.

Αφού τελειώσει η όλη διαδικασία, να γράφονται στην οθόνη οι ακέραιοι, οι οποίοι εξακολουθούν να υπάρχουν στην στοίβα.

Να υλοποιήσετε την στοίβα και με τις δύο λογικές που αναγράφονται πιο πάνω στις επεξηγήσεις.

```c
/* Α' τρόπος */
#include <stdio.h>
#define N 10

void push(int);
int pop();
```

```c
int stak[N];
int sp;

int main() {
    char ch;
    int ak;

    printf("GIVE CHARACTER ");
    scanf("%c", &ch);
    while (ch != 'E') {
        switch (ch) {
            // Push into stack
            case 'U':
                printf("GIVE int TO PUSH IN STACK\n");
                scanf("%d", &ak);
                push(ak);
                break;
            // Pop from stack
            case 'O':
                ak = pop();
                if (ak != -1)
                    printf ("%d\n", ak);
                break;
            default:
                printf ("WRONG CHARACTER\n");
        }
        fflush(stdin);
        printf("GIVE NEXT CHARACTER ");
        scanf("%c", &ch);
    }
    // Pop remaining numbers from stack
    while (sp != 0) {
        ak = pop ( );
        printf ("%5d", ak);
    }
    printf("\n");
    return 1;
}

void push (int ak) {
    if (sp == N) {
        printf ("STACK FULL. PUSH NOT ALLOWED\n");
        return;
    }
    else
        stak[sp++] = ak;
}
```

```c
int pop ( ) {
    if (sp == 0) {
        printf ("STACK EMPTY. POP NOT ALLOWED\n");
        return -1;
    }
    else
        return stak[--sp];
}


/* Β' Τρόπος */
#include <stdio.h>
#define N 10

void push(int);
int pop();
int stak[N];
int sp=-1;

int main() {
    char ch;
    int ak;

    printf("GIVE CHARACTER ");
    scanf("%c", &ch);
    while (ch != 'E') {
        switch (ch) {
            // Push into stack
            case 'U':
                printf("GIVE int TO PUSH IN STACK\n");
                scanf("%d", &ak);
                push(ak);
                break;
            // Pop from stack
            case 'O':
                ak = pop();
                if (ak != -1)
                    printf("%d\n", ak);
                break;
            default:
            printf ("WRONG CHARACTER\n");
        }
        fflush(stdin);
        printf("GIVE NEXT CHARACTER ");
        scanf("%c", &ch);
    }
    // Pop remaining numbers from stack
    while (sp != -1) {
        ak = pop();
```

```
            printf("%5d", ak);
        }
        printf("\n");
        return 1;
}

void push (int ak) {
        if (sp == N-1) {
            printf("STACK FULL. PUSH NOT ALLOWED\n");
            return;
        }
        else
            stak[++sp] = ak;
}

int pop () {
        if (sp == -1) {
            printf("STACK EMPTY. POP NOT ALLOWED\n");
            return -1;
        }
        else
            return stak[sp--];
}
```

Ουρές:

2. Να γραφεί ένα πρόγραμμα, στο οποίο να υλοποιήσετε μια ουρά αναμονής θετικών ακεραίων, με την χρήση πίνακα ακεραίων N θέσεων. Να γράψετε τις συναρτήσεις ώθησης (qinsert) και ανάκλησης (qremove) από την ουρά, στις οποίες να γίνεται έλεγχος πλήρους και κενής ουράς αντίστοιχα. Το πρόγραμμα να διαβάζει ένα χαρακτήρα από το πληκτρολόγιο, τον ch και:

- Εάν ο χαρακτήρας είναι το U να διαβάζετε ένα ακέραιο και να γίνεται ώθηση του ακεραίου στην ουρά.
- Εάν ο χαρακτήρας είναι το O να γίνεται ανάκληση ακεραίου από την ουρά, ο οποίος να εμφανίζεται στην οθόνη.
- Εάν ο χαρακτήρας είναι το E, να τελειώνει η όλη διαδικασία και να εμφανίζονται στην οθόνη τα περιεχόμενα της ουράς.
- Εάν ο χαρακτήρας δεν είναι κανείς από τους παραπάνω, να γράφεται στην οθόνη «ΛΑΘΟΣ ΧΑΡΑΚΤΗΡΑΣ» και να διαβάζεται νέος χαρακτήρας από το πληκτρολόγιο.

Αφού τελειώσει η όλη διαδικασία, να γράφονται στην οθόνη οι ακέραιοι, οι οποίοι εξακολουθούν να υπάρχουν στην ουρά.

Να εμπλουτίσετε το πρόγραμμά σας με μια συνάρτηση, η οποία θα μηδενίζει τις τιμές των σημείων εισόδου και εξόδου στοιχείων στην περίπτωση που συμπέσουν (πράγμα που σημαίνει κενή ουρά).

```c
#include <stdio.h>
#define N 5

void qinsert (int);
int qremove ();
int queue[N];
int ip, rp;

int  main() {
    char ch;
    int ak;

    printf("GIVE CHARACTER ");
    scanf("%c", &ch);
    while (ch != 'E') {
        switch (ch) {
            // Insert to queue
            case 'U':
                printf("GIVE int TO PUT IN QUEUE\n");
                scanf("%d", &ak);
                qinsert(ak);
                break;
            // Remove from queue
            case 'O':
                ak = qremove();
                if (ak != -1)
                    printf("%d\n", ak);
                break;
            default:
                printf("WRONG CHARACTER\n");
        }
        fflush(stdin);
        printf("GIVE NEXT CHARACTER ");
        scanf("%c", &ch);
    }
    // Remove remaining numbers from queue
    while (ip != rp) {
        ak = qremove();
        printf("%5d", ak);
    }
    printf("\n");
}

void  qinsert (int ak) {
```

```
    if (ip == N) {
        printf("QUEUE FULL. INSERT NOT ALLOWED\n");
        return;
    }
    else
        queue[ip++] = ak;
}

int  qremove () {
    if (ip == rp) {
        ip = 0;
        rp = 0;
        printf("QUEUE EMPTY. REMOVE NOT ALLOWED\n");
        return -1;
    }
    else
        return queue[rp++];
}
```

3. Να γραφεί ένα πρόγραμμα, το οποίο να υλοποιεί την μετατροπή από μεταδιατεταγμένη σε ενδοδιατεταγμένη παράσταση με τη χρήση στοίβας. Να διαβάζετε μια παράσταση με μορφή συμβολοσειράς, η οποία να περιέχει μονοψήφιους θετικούς ακέραιους αριθμούς και τελεστές + ή – ή * ή /. Π.χ., την παράσταση:

5 9 4 + - 7 3 * +

Να υπολογίζεται η ένθετη μορφή (άρα για την παραπάνω παράσταση το αποτέλεσμα, το οποίο είναι 13).

```
#include <stdio.h>
#define N 20
void push(int);
int pop();
int stak[N];
int sp;

int main() {
    char pin[N];
    char ch;
    int ak, tela, telb, k = 0, err = 0, apot;

    scanf("%s", pin);
    while (pin[k] != '\x0') {
        ch = pin[k];
        if (ch >= '0' && ch <= '9'){
```

```
            ak = ch - 48;
            push(ak);
        }
        else {
            tela = pop();
            telb = pop();
            switch(ch) {
                case '+':
                    push(telb + tela);
                    break;
                case '-':
                    push(telb - tela);
                    break;
                case '*':
                    push(telb  *tela);
                    break;
                case '/':
                    push(telb / tela);
                    break;
                default:
                    printf("WRONG STRING\n");
                    err = 1;
            }
            if (err == 1)
                break;
        }
        k++;
    }
    apot = pop();
    printf("\n%d\n", apot);
    printf("\n");
    return 1;
}

void push (int ak) {
    if (sp == N) {
        printf("STACK FULL. PUSH NOT ALLOWED\n");
        return;
    }
    else
        stak[sp++] = ak;
}

int pop () {
    if (sp == 0) {
        printf("STACK EMPTY. POP NOT ALLOWED\n");
        return -1;
    }
```

```
    else
        return stak[--sp];
}
```

ΕΡΓΑΣΤΗΡΙΟ 6

Α'. ΠΕΡΙΕΧΟΜΕΝΑ ΜΑΘΗΜΑΤΟΣ

Στο εργαστήριο 6 καλύπτονται τα παρακάτω θέματα:

- Απλά συνδεδεμένες λίστες. Δημιουργία.
- Λειτουργίες στις απλά συνδεδεμένες λίστες: αναζήτηση στοιχείου, εισαγωγή κόμβου, διαγραφή κόμβου, μετακίνηση κόμβου, αντιστροφή λίστας, συνένωση λιστών κλπ.

Β'. ΑΣΚΗΣΕΙΣ ΓΙΑ ΕΚΤΕΛΕΣΗ – ΕΠΙΔΕΙΞΗ / ΕΠΕΞΗΓΗΣΕΙΣ

1. Το πρόγραμμα που ακολουθεί ζητεί να δημιουργηθεί μια απλά συνδεδεμένη λίστα. Τα στοιχεία της λίστας είναι του είδους node και περιέχουν το όνομα ενός ατόμου, τον αριθμό μητρώου του και το οφειλόμενο σε αυτόν ποσόν. Στοιχεία εισάγονται στην λίστα μέχρι να δοθεί αριθμός μητρώου μηδέν ή αρνητικός. Το πρόγραμμα δεν προβλέπει το ενδεχόμενο να δοθεί ίδιος αριθμός μητρώου για δύο άτομα. Συνιστάται να το βελτιώσετε αργότερα (αφού κάνετε τις ασκήσεις), ώστε να αποκλείεται ένα τέτοιο ενδεχόμενο.

Μετά την καταχώρηση των στοιχείων, το πρόγραμμα, με τη χρήση της συνάρτησης display( ), εμφανίζει στην οθόνη τα ονόματα, τους αριθμούς μητρώου των ατόμων που έχουν καταχωρηθεί στην λίστα και το ποσόν που οφείλεται σε καθένα. Την συνάρτηση αυτή προφανώς να την χρησιμοποιείτε κάθε φορά που θέλετε να εμφανίσετε στην οθόνη τα περιεχόμενα της λίστας, δίνοντάς της ως όρισμα τον δείκτη στην κεφαλή της λίστας.

Μελετήστε το πρόγραμμα και αποθηκεύστε το, προκειμένου να εκτελέσετε και τις επόμενες ασκήσεις χρησιμοποιώντας το, χωρίς να ξαναδημιουργείτε την λίστα από την αρχή. Οι όποιες παρεμβάσεις σας προφανώς θα γίνονται αφού δημιουργηθεί η λίστα, άρα μετά την γραμμής 38 του προγράμματος.

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

struct node {
    char name[30];
    int am;
```

```
    float poson;
    struct node *next;
};

void display (struct node *);

int main (void) {
    struct node *head, *curr, *ptr;
    char nol[8];
    int num;

    /* Δημιουργία – Εισαγωγή στοιχείων στη λίστα */
    head = (struct node *) malloc(sizeof(struct node));
    curr = ptr = head;
    curr->next = NULL;
    printf("DWSTE ARI8MO MHTRWOY ");
    gets(nol);
    num = atoi(nol);

    while (num > 0) {
        curr->am = num;
        printf("DWSTE ONOMA ");
        gets(curr->name);
        printf("DWSTE POSON ");
        gets(nol);
        curr->poson = atof(nol);
        curr->next = (struct node *) malloc(sizeof(struct node));
        curr = curr->next;
        curr-
>next = NULL;
        printf("DWSTE EPOMENO ARI8MO MHTRWOY ");
        gets(nol);
        num = atoi(nol);
    }
    while (ptr->next != curr)
        ptr = ptr->next;
    ptr->next = NULL;
    free(curr);
    curr = NULL;
    display(head);
    system("pause");
    return 1;
}

void display (struct node *head) {
    struct node *curr;

    /* Εμφάνιση των στοιχείων της λίστας στην οθόνη */
```

```
    printf("\nEMFANISH TWN STOIXEIWN THS LISTAS\n");
    curr = head;
    while (curr != NULL) {
        printf ("ONOMA %s\t ARI8MOS MHTRWOY %5d\t POSON %10.2f\n", c
urr->name, curr->am, curr->poson);
        curr = curr->next;
    }
}
```

2. Να γραφεί μία συνάρτηση, η ofeiles( ), η οποία να αθροίζει τα ποσά που οφείλονται σε όλα τα άτομα της λίστας της άσκησης 1 και να υπολογίζει τον μέσο όρο των οφειλόμενων ποσών στα άτομα της λίστας. Η main( ) να γράφει στην οθόνη το συνολικό οφειλόμενο ποσόν και τον μέσο όρο των οφειλομένων ποσών ανά άτομο.

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

struct node {
    char name[30];
    int am;
    float poson;
    struct node *next;
};

void display (struct node *);
float ofeiles (struct node *, float *);

int main (void) {
    struct node *head, *curr, *ptr;
    char nol[8];
    int num;
    float mo, tot_poson;

    /* Δημιουργία – Εισαγωγή στοιχείων στη λίστα */
    head = (struct node *) malloc(sizeof(struct node));
    curr = ptr = head;
    curr->next = NULL;
    printf("DWSTE ARI8MO MHTRWOY ");
    gets(nol);
    num = atoi(nol);

    while (num > 0) {
        curr->am = num;
        printf("DWSTE ONOMA ");
        gets(curr->name);
```

```
            printf("DWSTE POSON ");
            gets(nol);
            curr->poson = atof(nol);
            curr->next = (struct node *) malloc(sizeof(struct node));
            curr = curr->next;
            curr-
>next = NULL;
            printf("DWSTE EPOMENO ARI8MO MHTRWOY ");
            gets(nol);
            num = atoi(nol);
        }
        while (ptr->next != curr)
            ptr = ptr->next;
        ptr->next = NULL;
        free(curr);
        curr = NULL;
        display(head);

        tot_poson = ofeiles(head, &mo);
        printf("\nSYNOLIKO OFEILOMENO POSON: %8.2f\n", tot_poson);
        printf("\nMESOS OROS OFEILOMENWN POSWN ANA ATOMO: %8.2f\n", mo);

        system("pause");
        return 1;
}

void display (struct node *head) {
        struct node *curr;

        /* Εμφάνιση των στοιχείων της λίστας στην οθόνη */
        printf("\nEMFANISH TWN STOIXEIWN THS LISTAS\n");
        curr = head;
        while (curr != NULL) {
            printf ("ONOMA %s\t ARI8MOS MHTRWOY %5d\t POSON %10.2f\n", c
urr->name, curr->am, curr->poson);
            curr = curr->next;
        }
}

float ofeiles (struct node *head, float *mo) {
        struct node *curr;
        int num = 0;
        float sum = 0;

        curr = head;
        while (curr != NULL) {
            num++;
            sum += curr->poson;
```

```
        curr = curr->next;
    }
    *mo = sum / num;
    return sum;
}
```

3. Να γραφεί μία συνάρτηση, η οποία να αναζητά ένα κόμβο στην λίστα της άσκησης 1, σύμφωνα με ένα όνομα που θα δίδεται στη main( ). Η συνάρτηση να επιστρέφει ένα δείκτη στον κόμβο που επισημάνθηκε το όνομα αυτό για πρώτη φορά. Εάν δεν υπάρχει το όνομα που ζητείται, η συνάρτηση να επιστρέφει NULL. Η main( ) να γράφει στην οθόνη το όνομα, τον αριθμό μητρώου του ατόμου και το ποσόν που οφείλεται σε αυτό (εφ' όσον το όνομα υπάρχει), αλλιώς να γράφει «ΑΝΥΠΑΡΚΤΟ ΟΝΟΜΑ».

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

struct node {
    char name[30];
    int am;
    float poson;
    struct node *next;
};

void display (struct node *);
struct node *find_name (struct node *, char *);

int main (void) {
    struct node *head, *curr, *ptr;
    char nol[8];
    int num;

    struct node * fptr;
    char onoma[30];

    /* Δημιουργία – Εισαγωγή στοιχείων στη λίστα */
    head = (struct node *) malloc(sizeof(struct node));
    curr = ptr = head;
    curr->next = NULL;
    printf("DWSTE ARI8MO MHTRWOY ");
    gets(nol);
    num = atoi(nol);
```

```
    while (num > 0) {
        curr->am = num;
        printf("DWSTE ONOMA ");
        gets(curr->name);
        printf("DWSTE POSON ");
        gets(nol);
        curr->poson = atof(nol);
        curr->next = (struct node *) malloc(sizeof(struct node));
        curr = curr->next;
        curr-
>next = NULL;
        printf("DWSTE EPOMENO ARI8MO MHTRWOY ");
        gets(nol);
        num = atoi(nol);
    }
    while (ptr->next != curr)
        ptr = ptr->next;
    ptr->next = NULL;
    free(curr);
    curr = NULL;
    display(head);

    printf("DWSTE ONOMA ");
    gets(onoma);
    fptr = find_name (head, onoma);
    if (fptr != NULL) {
        printf ("\nONOMA : %s\n", fptr->name);
        printf ("ARI8MOS MHTRWOY %5d\n", fptr->am);
        printf ("OFEILOMENO POSON %8.2f\n", fptr->poson);
    }
    else {
        printf("ANYPARKTO ONOMA\n");
    }



    system("pause");
    return 1;
}

void display (struct node *head) {
    struct node *curr;

    /* Εμφάνιση των στοιχείων της λίστας στην οθόνη */
    printf("\nEMFANISH TWN STOIXEIWN THS LISTAS\n");
    curr = head;
    while (curr != NULL) {
```

```
        printf ("ONOMA %s\t ARI8MOS MHTRWOY %5d\t POSON %10.2f\n", c
urr->name, curr->am, curr->poson);
        curr = curr->next;
    }
}

struct node *find_name (struct node *head, char *pin) {
    struct node *curr;
    int num = 0;
    float sum = 0;

    curr = head;
    while (curr != NULL) {
        if (strcmp(pin, curr->name) == 0)
            return curr;
        curr = curr->next;
    }
    return NULL;
}
```