2ο Project ΛΣ 2020-2021

Θεοδώρου Μιχαήλ 1067391
Φωτεινός Εμμανουήλ 1067428
Μαρκοδήμος Παναγιώτης 1067523

Μέρος 1

Ερώτημα Α)

Αρχικοποίηση μεταβλητών και shared memory, καθορισμός θέσης sum στην κοινή μνήμη και print αρχικής τιμής

```c
int main()
{
    //Αρχικοποίηση μεταβλητών και shared memory
    key_t shmkey;
    int pid[N];
    int *p;
    int shmid;
    int i;
    int child_status;

    shmkey = 9009;
    shmid = shmget (shmkey, N, 0644 | IPC_CREAT);


    if (shmid < 0)
    {
        perror ("shmget\n");
        exit (1);
    }

    p  = (int *) shmat (shmid, NULL, 0);

    //Καθορισμός θέσης αθροίσματος στο shared memory και εξίσωση με το 0
    int *sum_pointer = (p + N + 1);
    *sum_pointer = 0;
    printf(" Initial value of shared variable: %d\n\n", *p);
```

Γεμίζουμε το b+ με τιμές και δημιουργούμε τις διεργασίες-παιδιά, οι οποίες με θα μπουν στο σωστο if μέσω του pid και εκεί θα προσθέσουν την τιμή στο i τους στο sum, αφού την βρούν μέσω binary search

```
for (i=0; i< N; i++)        //Γεμιζουμε τον πινακα του b+ tree με i*2 τιμες σε καθε κελι i στην shared memory
{
*(p + i) = i*2;
printf("timh tou shared b+ tree %d osto keli: %d\n\n", i, *(p+i));
}

//Δημιουργία παιδιών
for (i=0; i<N; i++)
{
    pid[i] = fork();


    if (pid[i] < 0)
    {
        printf ("Fork error.\n");
        return (-1);
    }

    if (pid[i]==0)
    {
        //Κάθε παιδί τυπώνει το pid του και προσθέτει το i του στο sum, βρίσκωντας την τιμή του μέσω της δυαδικής αναζήτησης
        printf("I am a child (Id=%i)\n\n", getpid() );
        *sum_pointer = *sum_pointer + binarySearch(p,0,N-1,i);
        printf("I made the sum = %d \n", *sum_pointer);
        exit(i);
    }
}
```

Η διεργασία γονέας αναμένει να τελειώσουν όλα τα παιδιά και τυπώνει το πως τερματίστηκε το καθένα. Στην συνέχεια τυπώνει το τελικό άθροισμα, αποδεσμεύει την κοινή μνήμη και τερματίζει

```
    //Η διεργασία γονές περιμένει να τερματίσουν όλα τα παιδιά και τυπώνει τον τρόπο με τον οποίο τερματίστηκαν.
    for(i=0; i<N; i++)
    {
        pid_t wpid = waitpid(pid[i] ,&child_status, 0);
        if (WIFEXITED(child_status))
        {
            printf("Child%d terminated with exit status %d %d\n\n", pid[i], WEXITSTATUS(child_status), wpid);
        }
        else
        {
            printf("Child%d terminated abnormally\n\n", wpid);
        }
    }
    //Τύπωση τελικού αθροίσματος
    printf("sum = %d \n\n",*sum_pointer);
    shmdt(p); //Αποδέσμευση shared memory
    shmctl(shmid, IPC_RMID, 0);

    return(0);
}
```

Το αποτέλεσμα:

```
I made the sum = 6
I am a child (Id=8673)

I made the sum = 12
I am a child (Id=8674)

I made the sum = 20
I am a child (Id=8675)

I made the sum = 30
I am a child (Id=8676)

I made the sum = 42
I am a child (Id=8677)

I made the sum = 56
I am a child (Id=8678)

I made the sum = 72
I am a child (Id=8679)

I made the sum = 90
Child8670 terminated with exit status 0 8670

Child8671 terminated with exit status 1 8671

Child8672 terminated with exit status 2 8672

Child8673 terminated with exit status 3 8673

Child8674 terminated with exit status 4 8674

Child8675 terminated with exit status 5 8675

Child8676 terminated with exit status 6 8676

Child8677 terminated with exit status 7 8677

Child8678 terminated with exit status 8 8678

Child8679 terminated with exit status 9 8679

sum = 90
```

Ερώτημα Β)

Αρχικά, ορίζουμε τους σημαφόρους ως global, αρχικοποιούμε τις κατάλληλες μεταβλητές και το shared memory και ανοίγουμε τους σημαφόρους.

```c
int main(){
    //Αρχικοποίηση μεταβλητών και shared memory
    key_t shmkey;
    int pid[3];
    int *p;
    int shmid;
    int i;

    shmkey = 9090;
    shmid = shmget (shmkey, 3, 0644 | IPC_CREAT);

      if (shmid < 0)
    {
        perror ("shmget\n");
        exit (1);
    }

    p  = (int *) shmat (shmid, NULL, 0);

    printf(" Initial value of shared variable: %d\n\n", *p);

    //Άνοιγμα σεμαφόρων
    smoker0 = sem_open ("Sem0", O_CREAT | O_EXCL, 0644, 0);
    smoker1 = sem_open ("Sem1", O_CREAT | O_EXCL, 0644, 0);
    smoker2 = sem_open ("Sem2", O_CREAT | O_EXCL, 0644, 0);
    seller = sem_open ("Sem3", O_CREAT | O_EXCL, 0644, 0);
```

Στην συνέχεια, ορίζουμε τον seller=1 ώστε ο seller να τοποθετήσει υλικά στο άδειο τραπέζι και δημιουργούμε τους καπνιστές ως διεργασίες-παιδιά

```c
//seller = 1 για να τοποθετήσει τα πρώτα υλικά ο seller στο τραπέζι
sem_post(seller);

//Δημιουγία καπνιστών
for (i=0; i<3; i++)
{
    pid[i] = fork();


    if (pid[i] < 0)
    {
        printf ("Fork error.\n");
        return (-1);
    }
}
```

Δημιουργούμε τα ifs μεσα στα οποία θα τρέχει η κάθε διεργασία-παιδι καπνιστής με χρήση του pid. Εκεί, θα ξυπνάει το κάθε καπνιστή ο seller όταν τοποθετήσει τα προϊόντα που χρειάζεται, αυτός στην συνέχεια θα τα παίρνει και θα ξυπνάει τον seller.

```
//Άν είσαι ο καπνιστής 1
if (pid[0] == 0)
{
while(1 != 0){
    //Αναμονή για αφύπνιση από seller αν τοποθετηθούν τα υλικά που του λείπουν
    sem_wait (smoker0);
    printf (" Eimai o kapnisths 1 kai xreiazomai xarti kai spirta \n\n");
    TakeMaterialsFromTable(p, 0);  //Παίρνει τα υλικά
    sem_post (seller); //Αφύπνιση seller
}
}
else if (pid[0] != 0 && pid[1] == 0) //Άν είσαι ο καπνιστής 2
{
while(1 != 0){
   sem_wait(smoker1);
   printf (" Eimai o kapnisths 2 kai xreiazomai kapno kai spirta \n\n");
   TakeMaterialsFromTable(p, 1);
   sem_post (seller);
}
}
else if (pid[0] != 0 && pid[1] != 0 && pid[2] == 0) //Άν είσαι ο καπνιστής 3
{
while(1 != 0)
{
   sem_wait (smoker2);
   printf (" Eimai o kapnisths 3 kai xreiazomai kapno kai xarti \n\n");
   TakeMaterialsFromTable(p, 2);
   sem_post (seller);
}
}
```

Η διεργασία γονέας είναι ο seller και αποφασίζει μέσω ενός random generator, την τιμη του τρέχων χρόνου και του i, ποιά προϊόντα θα τοποθετήσει στο τραπέζι και ξυπνάει τον κατάλληλο smoker. Αυτό το loop εκτελείται 10 φορές, ώστε να τελειώνει κάποτε το πρόγραμμα, αλλα θα μπορούσαμε θέσουμε την συνθήκη ώστε να εκτελείται άπειρες.

```
//Αρχικοποίηση r για χρήση στην τυχαία επιλογή
int r = 0;
for (i = 0; i < 10; i++)
{
    sem_wait(seller); //Αναμόνη για αφύπνιση απο smoker (με εξαίρεση την πρώτη φόρα που θα τρέξει)
    srand(time(NULL) + i); //Τυχαία επιλογή μέσω της τιμής του τρέχων χρόνου και του i
    r = rand() % 3; //mod 3 γιατί έχουμε 3 επιλογές
    printf("o pwlhths topothetei ta proinda ston pagko: \n\n");
    DecideWhichMaterialsToSell(p, r); //Τοποθέτηση προιόντων
    //Αφύπνιση κατάλληλου smoker
    if (r == 0)
        sem_post (smoker0);
    else if (r == 1)
        sem_post (smoker1);
    else
        sem_post (smoker2);
    }
```

Μετά των απαιτούμενο αριθμό εκτελέσεων του seller, οι σημαφόροι κλείνουν, η κοινή μνήμη αποδεσμεύεται και ο seller τερματίζει και μαζί και οι διεργασίες-παιδιά καπνιστές.

```
    //Κλείσιμο σεμαφόρων
    sem_unlink ("Sem0");
    sem_close(smoker0);
    sem_unlink ("Sem1");
    sem_close(smoker1);
    sem_unlink ("Sem2");
    sem_close(smoker2);
    sem_unlink ("Sem3");
    sem_close(seller);

    shmdt(p); //Αποδέσμευση shared memory
    shmctl(shmid, IPC_RMID, 0);

return(0);
}
```

Τα functions που χρησιμοποιήθηκαν:

```c
//Τύπωση τραπεζιού και προιόντων που περιέχει επάνω σχηματικά.
void print_Table(int n, int m, int kapnos, int xarti, int spirta)
{
    int i, j;
    for (i = 1; i <= n; i++)
    {
        for (j = 1; j <= m; j++)
        {
            if ((i==1 || i==n || j==1 || j==m) && i != 2)
                printf("*");
            else if (i==2 && j==2)
                printf("KAPNOS");
            else if (i==2 && j==10)
                printf("XARTI");
            else if (i==2 && j==17)
                printf("SPIRTA");
            else if ((i==3 || j==11 || j==23) && i != 2)
                printf("*");
            else if (i==4 && j==4)
                printf("%d", kapnos);
            else if (i==4 && j==16)
                printf("%d", xarti);
            else if (i==4 && j==29)
                printf("%d", spirta);
            else
                printf(" ");
        }
        printf("\n");
    }

}
```

```c
//Function όπου χρησιμοποιεί ο καπνιστής για να πάρει υλικά απο το τραπέζι ανάλογα με το τι υπάρχει, το οποίο καθορίζεται απο το
int TakeMaterialsFromTable(int *p, int r)
{   //Θέση υλικού στο shared memory
    int *kapnos = p;
    int *xarti = (p+1);
    int *spirta = (p+2);
    //Τι υλίκο παίρνει
    if (r==0){
        *xarti = 0;
        *spirta = 0;
    }
    else if(r==1) {
        *kapnos = 0;
        *spirta = 0;
    }
    else {
        *kapnos = 0;
        *xarti =0;
    }
}
```

```c
//Function όπου χρησιμοποιεί ο πωλητής για να βάλει υλικά στο τραπέζι, το οποίο καθορίζεται απο το r
int DecideWhichMaterialsToSell(int *p, int r)
{   //Θέση υλικού στο shared memory
    int *kapnos = p;
    int *xarti = (p + 1);
    int *spirta = (p + 2);
    //Τι υλίκο βάζει
    if (r==0){
        *kapnos = 0;
        *xarti = 1;
        *spirta = 1;
        print_Table(7, 38, *p, *(p+1), *(p+2));
    }
    else if(r==1) {
        *kapnos = 1;
        *xarti = 0;
        *spirta = 1;
        print_Table(7, 38, *p, *(p+1), *(p+2));
    }
    else {
        *kapnos = 1;
        *xarti = 1;
        *spirta = 0;
        print_Table(7, 38, *p, *(p+1), *(p+2));
    }
}
```

Το αποτέλεσμα:

```
**************************************
 KAPNOS        XARTI       SPIRTA
**************************************
*  1       *    0       *     1        *
*          *            *              *
*          *            *              *
**************************************
 Eimai o kapnisths 2 kai xreiazomai kapno kai spirta

o pwlhths topothetei ta prointa ston pagko:

**************************************
 KAPNOS        XARTI       SPIRTA
**************************************
*  1       *    1       *     0        *
*          *            *              *
*          *            *              *
**************************************
 Eimai o kapnisths 3 kai xreiazomai kapno kai xarti

o pwlhths topothetei ta prointa ston pagko:

**************************************
 KAPNOS        XARTI       SPIRTA
**************************************
*  1       *    1       *     0        *
*          *            *              *
*          *            *              *
**************************************
 Eimai o kapnisths 3 kai xreiazomai kapno kai xarti
```

Ερώτημα Γ)

Χρησιμοποιούμε δυο σημαφόρους.

Αρχικά, ορίζουμε τους σημαφόρους ως global, δημιοργούμε πίνακες για την αποθήκευση του pid των P,Q και όλων των διεργασιών E, μια μεταβλήτη για να αποθηκεύει την κατάσταση της διεργασίας-παιδιού, και ανοίγουμε τους σεμαφόρους

```c
//Ορισμός σημαφόρων ως global
Semaphore *s0;
Semaphore *s1;

int main()
{
pid_t pid_pq[2]; //Πίνακας αποθήκευσης pid διεργασιών P και Q
pid_t pid[10]; //Πίνακας αποθήκευσης pid διεργασιών παιδιών των P και Q
int child_status;

//Άνοιγμα σεμαφόρων
s0 = sem_open ("Sem1", O_CREAT | O_EXCL, 0644, 0);
s1 = sem_open ("Sem2", O_CREAT | O_EXCL, 0644, 0);
```

Στην συνέχεια, μέσω ifs και την χρήση των pid, καθορίζουμε την λειτουργία κάθε διεργασίας. Η P δημιουργεί τις E1, E5, E8, E9, E7, περιμένωντας να τελείωσει η λειτουργία της προηγούμενης πριν ξεκινήσει η επόμενη και τυπώνοντας τον τρόπο με τον οποίο τερματίστηκε η κάθε μία. Επίσης, έχουν τοποθετηθεί οι 2 σημαφόροι στα κατάλληλα σημεία, όπως μετά το τέλος της E1, ώστε να εκτελούνται οι διεργασίες που ανήκουν στην Q και όχι στην P και το ανάποδο με την απαιτούμενη σειρά.

```
//Αν είσαι η διεργασία P
if (pid_pq[0] == 0)
{
    //Ξεκινά την διεργασία E1
    pid[1] = fork();
    //Αν είσαι η διεργασία E1
    if (pid[1] == 0)
    {
        system("ls -l"); //Εκτελεση εντολής συστήματος και έξοδος
        exit(1);
    }
    waitpid(pid[1], &child_status, 0); //Αναμονή μέχρι να τελείωσει η E1 ώστε να ξεκινήσει η επόμενη, η E5, σύμφωνα με την λειτουργία του begin;
    if (WIFEXITED(child_status)) //If για την εκτύπωση του πως τερματίστηκε η E1
    {
        printf("Process%d terminated with exit status %d %d\n", pid[1], WEXITSTATUS(child_status), pid[1]);
    }
    else
    {
        printf("Process%d terminated abnormally\n", pid[1]);
    }
    //Το πρόγραμμα συνεχίζει ακριβώς με τον ίδιο τρόπο όπως το παραπάνω κομμάτι με εξαίρεση τους σημαφόρους όπου υπάρχουν σχόλια.
    sem_post(s0); //s0 γίνεται 1 ώστε να μπορεί να ξεκινήσει η E2 αφού πλέον τελείωσε η E1
```

Η εκτέλεση Q και των υπόλοιπων E λειτουργεί με τον ίδιο τρόπο.

Στο τέλος:

```
//Τύπωση τερματισμού P και Q
waitpid(pid_pq[0], &child_status, 0);
    if (WIFEXITED(child_status))
    {
        printf("Process P terminated with exit status %d %d\n", WEXITSTATUS(child_status), pid_pq[0]);
    }
    else
    {
        printf("Process P terminated abnormally\n");
    }

waitpid(pid_pq[1], &child_status, 0);
    if (WIFEXITED(child_status))
    {
        printf("Process Q terminated with exit status %d %d\n", WEXITSTATUS(child_status), pid_pq[1]);
    }
    else
    {
        printf("Process Q terminated abnormally\n");
    }

//Κλείσιμο σεμαφόρων
sem_unlink("Sem1");
sem_close(s0);
sem_unlink("Sem2");
sem_close(s1);

printf("My job here is done.\n");
return (0);
}
```

Η αρχική διεργασία αναμένει να τελειώσουν οι P και Q και κλείνει τους σημαφόρους.

Το αποτέλεσμα:

```
-rwxr-xr-x 1 dusk dusk 17216 Feb  2 00:01 1A
-rw-r--r-- 1 dusk dusk  2069 Feb  2 00:01 1A.c
-rwxr-xr-x 1 dusk dusk 17656 Feb  2 01:10 1B
-rw-r--r-- 1 dusk dusk  4260 Feb  1 23:57 1B.c
-rwxr-xr-x 1 dusk dusk 17248 Jan 27 20:15 1C
-rw-r--r-- 1 dusk dusk  4603 Jan 27 20:15 1C.c
-rwxr-xr-x 1 dusk dusk 16920 Jan  8 19:27 ero1
-rw-r--r-- 1 dusk dusk   694 Feb  1 23:59 ero1.c
-rwxr-xr-x 1 dusk dusk 17048 Jan 10 20:41 erob
-rwxr-xr-x 1 dusk dusk 17224 Jan 10 22:23 meros1_b
-rw-r--r-- 1 dusk dusk  5486 Jan 10 22:23 meros1_b.c
Process7417 terminated with exit status 7 7417
Process P terminated with exit status 10 7394
total 68
-rwxr-xr-x 1 dusk dusk 17216 Feb  2 00:01 1A
-rw-r--r-- 1 dusk dusk  2069 Feb  2 00:01 1A.c
-rwxr-xr-x 1 dusk dusk 17656 Feb  2 01:10 1B
-rw-r--r-- 1 dusk dusk  4260 Feb  1 23:57 1B.c
-rwxr-xr-x 1 dusk dusk 17248 Jan 27 20:15 1C
-rw-r--r-- 1 dusk dusk  4603 Jan 27 20:15 1C.c
-rwxr-xr-x 1 dusk dusk 16920 Jan  8 19:27 ero1
-rw-r--r-- 1 dusk dusk   694 Feb  1 23:59 ero1.c
-rwxr-xr-x 1 dusk dusk 17048 Jan 10 20:41 erob
-rwxr-xr-x 1 dusk dusk 17224 Jan 10 22:23 meros1_b
-rw-r--r-- 1 dusk dusk  5486 Jan 10 22:23 meros1_b.c
Process7420 terminated with exit status 4 7420
Process Q terminated with exit status 11 7396
My job here is done.
```

Μέρος 2

Ερώτημα Α)

Χρονική Στιγμή	Άφιξη	Εικόνα Μνήμης	Ουρά Μνήμης	ΚΜΕ	Ε/Ε	Ουρά ΚΜΕ	Ουρά Ε/Ε	Τέλος
0	P1	<Οπή 2MB>	P1	-	-	-	-	-
1	Q1	<P1> <Οπή 1748K>	Q1	P1	-	-	-	-
2	P2	<P1> <Q1> <Οπή 548K>	P2	P1	-	Q1	-	-

3	Q2	<P1> <Q1> <P2> <Οπή 248K>	Q2	P1	-	Q1,P2	-	-
4	P3	<P1> <Q1> <P2> <Οπή 248K>	Q2,P3	P1	-	Q1,P2	-	-
5	-	<P1> <Q1> <P2> <Οπή 248K>	Q2,P3	Q1	-	P2,P1	-	-
6	-	<P1> <Q1> <P2> <Οπή 248K>	Q2,P3	P2	Q1	P1	-	-
7	-	<P1> <Q1> <P2> <Οπή 248K>	Q2,P3	P2	Q1	P1	-	-
8	-	<P1> <Q1> <P2> <Οπή 248K>	Q2,P3	P2	Q1	P1	-	-
9	-	<P1> <Q1> <P2> <Οπή 248K>	Q2,P3	P2	-	P1,Q1	-	-
10	-	<P1> <Q1> <P2> <Οπή 248K>	Q2,P3	P1	-	Q1,P2	-	-
11	-	<P1> <Q1> <P2> <Οπή 248K>	Q2,P3	P1	-	Q1,P2	-	-
12	-	<P1> <Q1> <P2> <Οπή 248K>	Q2,P3	P1	-	Q1,P2	-	-
13	-	<P1> <Q1> <P2> <Οπή 248K>	Q2,P3	P1	-	Q1,P2	-	P1
14	-	<Q1>	P3	Q1	-	P2,Q2	-	-

		<P2> <Q2> <Οπή 48K>						
15	-	<Q1> <P2> <Q2> <Οπή 48K>	P3	P2	Q1	Q2	-	-
16	-	<Q1> <P2> <Q2> <Οπή 48K>	P3	P2	Q1	Q2	-	-
17	-	<Q1> <P2> <Q2> <Οπή 48K>	P3	P2	Q1	Q2	-	Q1
18	-	<P2> <Q2> <P3> <Οπή 548K>	-	P2	-	Q2,P3	-	P2
19	-	<Q2> <P3> <Οπή 848K>	-	Q2	-	P3	-	-
20	-	<Q2> <P3> <Οπή 848K>	-	P3	Q2	-	-	-
21	-	<Q2> <P3> <Οπή 848K>	-	P3	Q2	-	-	-
22	-	<Q2> <P3> <Οπή 848K>	-	P3	Q2	-	-	-
23	-	<Q2> <P3> <Οπή 848K>	-	P3	-	Q2	-	-
24	-	<Q2> <P3> <Οπή 848K>	-	Q2	-	P3	-	-
25	-	<Q2> <P3> <Οπή 848K>	-	P3	Q2	-	-	-
26	-	<Q2> <P3> <Οπή 848K>	-	P3	Q2	-	-	-

27	-	<Q2> <P3> <Οπή 848K>	-	P3	Q2	-	-	Q2
28	-	<P3> <Οπή 1348K>	-	P3	-	-	-	P3

<u>Ερώτημα Β)</u>

α)Με βάση τα δεδομένα της εκφωνησης:

- Εφόσον το μέγεθος σελίδας είναι ίσο με 2^10 άρα η μετατόπιση αποτελείται από 10 bits (k=10).
- ‘Επειτα ο πίνακας σελίδων αποτελείται από 256 = 2^8 εγγραφές , αυτό σημαίνει ότι τα bit του αριθμού ιδεατής σελίδας ισούνται με 8(n-κ=8) . Συνεπώς ολόκληρη η λογική διεύθυνση έχει 18 bit (n = 18)
- Στη συνέχεια , μας παρέχεται η πληροφορία πως η ο αριθμός πλαισίων αποτελείται από 1024 θέσεις = 2^10 άρα 10 bits. Έστω m τα bit της φυσικής διεύθυνσης. Με k=10 τα bits της μετατόπισης , έχουμε m = 20 bit (και m - k = 10).

Συμπεραίνοντας :
Για την αναπαράσταση κάθε λογικής διεύθυνσης χρειάζονται 18 bits (n=18).
Για την αναπαράσταση κάθε φυσικής διεύθυνσης χρειάζονται 20 bits (m=20).

β)
Η <u>λογική διεύθυνση 0A0A</u> στο δεκαεξαδικό αντιστοιχεί στο δυαδικό:

00 0000 1010 0000 1010

όμως εφόσον τα bit μετατόπισης ισούνται με 10
(k = 10) και αυτα βρισκονται στο τελος της διεύθυνσης απομενουν τα 8 bit του <u>αριθμου σελιδας</u>:

00 0000 10 = 02 (στο δεκαεξαδικο**)**

Με βάση τον Πίνακα Σελίδων η λογική σελίδα 02 αντιστοιχεί στην φυσική σελιδα (πλαίσιο) **20C**
που αντιστοιχεί σε **0010 0000 1100** που γίνεται **10 0000 1100** στα 10 bits του αριθμού πλαισίου.
Συνεπώς η <u>τελική φυσικη διευθυνση</u> ειναι:

αριθμός πλαισίου + μετατόπιση =
1000 0011 0010 0000 101

Ακολουθία:	2	5	8	1	8	7	5	1	8	2	4	2	1	3	6	4	7	5	3	7
Χρονική Στιγ:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	2	2	2	2	2	7	7	7	7	2	2	2	2	2	2	4	4	4	4	4
1		5	5	5	5	5	5	5	5	5	4	4	4	4	6	6	6	6	3	3
2			8	8	8	8	8	8	8	8	8	8	8	3	3	3	3	5	5	5
3				1	1	1	1	1	1	1	1	1	1	1	1	1	7	7	7	7

Σφάλματα σελίδας: 9