

# 1η Εργαστηριακή Άσκηση (Project)

Λειτουργικά Συστήματα  
2024 - 2025

## Ονοματα:

Μαυριδης Κωνσταντinos, 1100620, up1100620@ac.upatras.gr

Μήτσαινας Μιχαήλ-Χαράλαμπος, 1100628, [st1100628@ceid.upatras.gr](mailto:st1100628@ceid.upatras.gr)

Λουκανάρης Κωνσταντίνος, 1100610, up1100610@ac.upatras.gr

Εχουν υλοποιηθεί όλα τα τμήματα του project και δουλεύουν όλα τα ζητούμενα κομμάτια κωδικα. Συγκεκριμένα έχει υλοποιηθεί :

- ☒ Άσκηση 1
- ☒ Άσκηση 2
- ☒ Άσκηση 3
- ☒ Άσκηση 4

## Άσκηση 1

Για την ολοκλήρωση του shell script χρησιμοποιήσαμε πολλά από αυτά που μάθαμε κατά τη διάρκεια του εξαμήνου και υλοποιήσαμε όλες τις λειτουργίες με το εξής σκεπτικό:

### [Αναγνώριση λειτουργίας]

Αρχικά διαβάζοντας το 1ο semantic value από το input στο command line εκτελούμε και την ανάλογη διεργασία. Με την εκτέλεση `./processes_ipc.sh X` όπου X είναι το 1ο semantic value, εκτελεί ανάλογα εισαγωγή, αναζήτηση, ενημέρωση, εμφάνιση ή ανάλυση και αποθήκευση στατιστικών.

```
if [ $1 == "insert" ]; then
    insert_data
```

```

elif [ $1 == "search" ]; then
    search_passenger
elif [ $1 == "update" ]; then
    update_passenger "$2" "$3" "$4"
elif [ $1 == "display" ]; then
    display_file
elif [ $1 == "reports" ]; then
    generate_reports
else
    echo "Type one of the following functions: insert, search, update, display, reports"
fi

```

[1]

## “Εισαγωγή-insert”

Ξεκινώντας, όπως ζητείται, δίνουμε την επιλογή να δώσει ο χρήστης file path ως είσοδο, αλλιώς αν πατήσει Enter (δηλ. Το filepath μένει κενό -> -z) τότε παίρνει ως path το predefined αρχείο passenger\_data.csv. Έπειτα του δίνεται η σωστή δομή μιας νέας εγγραφής όπως περιγράφεται στην εκφώνηση. Αφού τσεκάρουμε αν κάθε field έχει σωστό τύπο/δομή τότε αλλάζουμε τα “,” με “;” διότι τα csv χρησιμοποιούν “;” ως διαχωριστικό ενώ ο χρήστης έβαλε “,” και το αλλαγμένο record γράφεται στο csv του αντίστοιχου file path. Ο χρήστης μπορεί να συνεχίσει να βάζει εγγραφές μέχρι να πατήσει ctrl+c και να βγεί.

```

insert_data() {
    echo "Insert data:"
    read -p "Enter the filename (or press Enter to input manually): " filename

    if [[ -z "$filename" ]]; then
        echo "Enter passenger data in the format: [code],[fullname],[age],[country],[status (Passenger or Crew)],[rescued (Yes or No)]:"
        while read -r line; do
            if [[ -z "$line" ]]; then
                break
            fi

            if [[ "$line" =~ ^[A-Za-z0-9]+,[A-Za-z]+[[:space:]]+[A-Za-z]+,[0-9]+,[^,]+,(Passenger|Crew),(Yes|No)$ ]];
            then

```

```

        echo "$line" | tr ',' ';' >> "$FILE"
    else
        echo "Invalid format: $line"
    fi
done
else
    if [[ -f "$filename" ]]; then
        echo "Enter passenger data in the format: [code],[fullname],[age],[country],[status
(Passenger or Crew)],[rescued (Yes or No)]:"
        while read -r line; do
            if [[ -z "$line" ]]; then
                break
            fi

            if [[ "$line" =~
^[A-Za-z0-9]+,[A-Za-z]+[:space:]+[A-Za-z]+,[0-9]+,[^,]+,(Passenger|Crew),(Yes|No)$ ]];
then
                echo "$line" | tr ',' ';' >> "$filename"
            else
                echo "Invalid format: $line"
            fi
        done
    else
        echo "Error: File '$filename' not found."
    fi
fi
}

```

**Παράδειγμα εκτέλεσης εισαγωγής του “1286,Michael Mitsainas,20,Greece,Passenger,Yes”:**

```

1282 1285;Kostas Loukanaris;21;Greece;Passenger;Yes
1283 1286;Michael Mitsainas;20;Greece;Passenger;Yes
1284
PROBLEMS OUTPUT DEBUG CONSOLE PORTS GITLENS SQL CONSOLE COMMENTS TERMINAL
soktas@matebook:~/Documents/UniDocuments/Third Year/Operating Systems/CEID-OS-Projects/Project1/askhsh1$ ./processes_ipc.sh insert
Insert data:
Enter the filename (or press Enter to input manually):
Enter passenger data in the format: [code],[fullname],[age],[country],[status (Passenger or Crew)],[rescued (Yes or No)]:
1286,Michael Mitsainas,20,Greece,Passenger,Yes
^C

```

[2]

“Αναζήτηση-search”

Αφού λοιπόν δώσει ο χρήστης το όνομα ή το επίθετο κάποιου, τότε με τη χρήση της awk τσεκάρουμε όλο το csv του αρχείου που θέλουμε διαχωριζόμενο είτε από “;” είτε από “,” για να βρούμε ταίρι του “name” που έδωσε ο χρήστης. Το ψάξιμο γίνεται στη δεύτερη στήλη του csv (\$2)

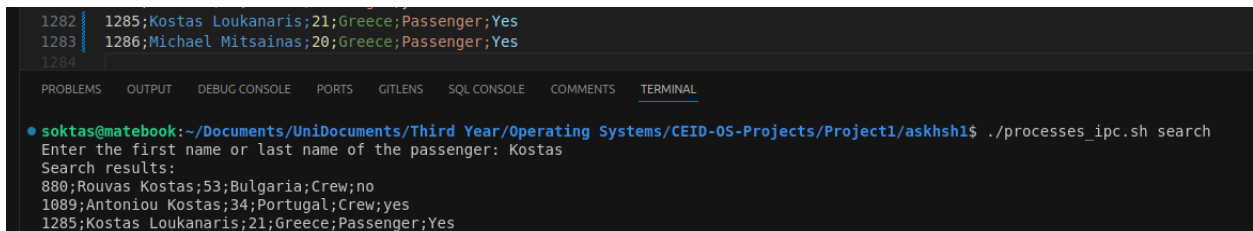
είτε στα ονόματα (names[1]) είτε στα επίθετα (names[2]) με case insensitive τρόπο. Αν βρεθεί κάτι εκτυπώνεται όλη η γραμμή (results=print \$0).

```
search_passenger() {
    read -p "Enter the first name or last name of the passenger: " name

    results=$(awk -F'[;,]' -v search="$name" '{
        split($2, names, " ");
        if (tolower(names[1]) == tolower(search) || tolower(names[2]) == tolower(search)) {
            print $0
        }
    }' $FILE)

    if [[ -n "$results" ]]; then
        echo "Search results:"
        echo "$results"
    else
        echo "Passenger not found."
    fi
}
```

**Παράδειγμα εκτέλεσης αναζήτησης του ονόματος “Kostas”:**



```
1282 | 1285;Kostas Loukanaris;21;Greece;Passenger;Yes
1283 | 1286;Michael Mitsalinas;20;Greece;Passenger;Yes
1284 |
PROBLEMS OUTPUT DEBUG CONSOLE PORTS GIT LENS SQL CONSOLE COMMENTS TERMINAL
• soktas@matebook: ~/Documents/UniDocuments/Third Year/Operating Systems/CEID-OS-Projects/Project1/askhsh1$ ./processes_ipc.sh search
Enter the first name or last name of the passenger: Kostas
Search results:
880;Rouvas Kostas;53;Bulgaria;Crew;no
1089;Antoniou Kostas;34;Portugal;Crew;yes
1285;Kostas Loukanaris;21;Greece;Passenger;Yes
```

[3]

“Ενημέρωση-update”

Καταρχάς στην λειτουργία για ενημέρωση διαβάζονται 2 πράγματα, το “κλειδί” της εγγραφής που πάμε να ενημερώσουμε (\$1) και το πεδίο/όλη η εγγραφή που θέλουμε να αλλάξουμε και τα πάντα μετά το “:” ({@:\$2}). Μετά με χρήση της grep ψάχνουμε να βρούμε αν όντως υπάρχει αυτή η εγγραφή που πάμε να αλλάξουμε. Αφού το βρούμε πάμε και διαχωρίζουμε αυτό το string που πήρε το {@:\$2} σε πεδίο και λοιπά με διαχωριστικό

το ":". Αν το πεδίο είναι "record" τότε πρέπει όπως στην insert να βάλει ο χρήστης σωστά όλη τη νέα εγγραφή, αλλιώς αν έδωσε συγκεκριμένο πεδίο/στήλη τότε με μια απλή switch case αλλάζει μόνο η τιμή αυτού και όλες οι άλλες μένουν ίδιες.

```
update_passenger() {
    person_to_search="$1"
    fieldnvalue="${@:2}" #ta pairnei ola meta to :

    if [ -z "$filepath" ]; then
        filepath="./passenger_data.csv"
    elif [ ! -f "$filepath" ]; then
        echo "$filepath doesn't exist"
        return
    fi

    match=$(grep -i "$person_to_search" "$filepath")

    if [ -z "$match" ]; then
        echo "No match found for '$person_to_search'."
        return
    else
        echo "Matches:"
        echo "$match"
    fi

    IFS=":" read -r field value <<< "$fieldnvalue"

    if [ -z "$field" ] || [ -z "$value" ]; then
        echo "Give correct input format"
        return
    fi

    if [ "$field" == "record" ]; then
        if [ [ "$value" =~
^[0-9]+,[A-Za-z]+[[:space:]]+[A-Za-z]+,[0-9]+,[^,]+,(Passenger|Crew),(Yes|No)$ ] ]; then

            value=$(echo "$value" | tr ',' ';')

            sed -i "s|$match|$value|" "$filepath"
            echo "New record: $value"
            echo "Old record: $match"
            echo "Full record update successful"
        else
            echo "Invalid input format"
        fi
    else
        IFS=";" read -r code fullname age country status rescued <<< "$match"
```

```

    case "$field" in
        code) code="$value" ;;
        fullname) fullname="$value" ;;
        age) age="$value" ;;
        country) country="$value" ;;
        status) status="$value" ;;
        rescued) rescued="$value" ;;
        *)
            echo "Invalid field. Please use a valid field name."
            return
            ;;
    esac

    new_record="$code;$fullname;$age;$country;$status;$rescued"

    sed -i "s|$match|$new_record|" "$filepath"
    echo "New record: $new_record"
    echo "Old record: $match"
    echo "Field update successful"

fi
}

```

**Παράδειγμα εκτέλεσης ενημέρωσης της εγγραφής “1286,Michael Mitsainas,20,Greece,Passenger,Yes” σε “1286,Mixalis Mitsainas,21,Greece,Crew,No”:**

```

1282 | 1285;Kostas Loukanaris;21;Greece;Passenger;Yes
1283 | 1286;Mixalis Mitsainas;21;Greece;Crew;No
1284
PROBLEMS OUTPUT DEBUG CONSOLE PORTS GIT LENS SQL CONSOLE COMMENTS TERMINAL
● soktas@matebook: ~/Documents/UniDocuments/Third Year/Operating Systems/CEID-05-Projects/Project1/askhsh1$ ./processes_ipc.sh update 1286 record:1286,Mixalis Mitsainas,21,Greece,Crew,No
Matches:
1286;Michael Mitsainas;20;Greece;Passenger;Yes
New record: 1286;Mixalis Mitsainas;21;Greece;Crew;No
Old record: 1286;Michael Mitsainas;20;Greece;Passenger;Yes
Full record update successful

```

**Παράδειγμα ενημέρωσης του ονόματος του πάλι σε “Michael”:**

```

1282 | 1285;Kostas Loukanaris;21;Greece;Passenger;Yes
1283 | 1286;Michael Mitsainas;21;Greece;Crew;No
1284
PROBLEMS OUTPUT DEBUG CONSOLE PORTS GIT LENS SQL CONSOLE COMMENTS TERMINAL
● soktas@matebook: ~/Documents/UniDocuments/Third Year/Operating Systems/CEID-05-Projects/Project1/askhsh1$ ./processes_ipc.sh update 1286 fullname:Michael Mitsainas
Matches:
1286;Mixalis Mitsainas;21;Greece;Crew;No
New record: 1286;Michael Mitsainas;21;Greece;Crew;No
Old record: 1286;Mixalis Mitsainas;21;Greece;Crew;No
Field update successful

```

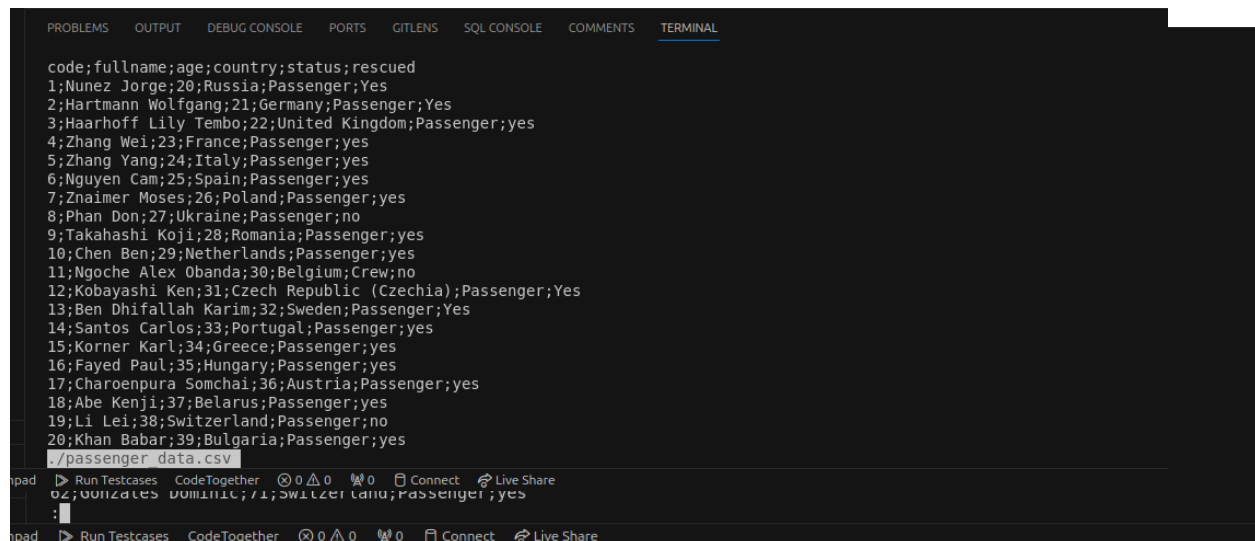
[ 4 ]

“Εμφάνιση-display”

Εδώ αξιοποιούμε πλήρως την εντολή “less” η οποία by default προχωράει πιο κάτω στο τερματικό με το πάτημα του spacebar. Για έξοδο πατήστε “q”

```
display_file() {  
    filepath="$1"  
  
    if [ -z "$filepath" ]; then  
        filepath="./passenger_data.csv"  
    elif [ ! -f "$filepath" ]; then  
        echo "$filepath doesn't exist"  
        return  
    fi  
  
    less "$filepath"  
}
```

Παράδειγμα εμφάνισης δεδομένων:



```
code;fullname;age;country;status;rescued  
1;Nunez Jorge;20;Russia;Passenger;Yes  
2;Hartmann Wolfgang;21;Germany;Passenger;Yes  
3;Haarhoff Lily Tembo;22;United Kingdom;Passenger;yes  
4;Zhang Wei;23;France;Passenger;yes  
5;Zhang Yang;24;Italy;Passenger;yes  
6;Nguyen Cam;25;Spain;Passenger;yes  
7;Znaimer Moses;26;Poland;Passenger;yes  
8;Phan Don;27;Ukraine;Passenger;no  
9;Takahashi Koji;28;Romania;Passenger;yes  
10;Chen Ben;29;Netherlands;Passenger;yes  
11;Ngoche Alex Obanda;30;Belgium;Crew;no  
12;Kobayashi Ken;31;Czech Republic (Czechia);Passenger;Yes  
13;Ben Dhifallah Karim;32;Sweden;Passenger;Yes  
14;Santos Carlos;33;Portugal;Passenger;yes  
15;Korner Karl;34;Greece;Passenger;yes  
16;Fayed Paul;35;Hungary;Passenger;yes  
17;Charoenpura Somchai;36;Austria;Passenger;yes  
18;Abe Kenji;37;Belarus;Passenger;yes  
19;Li Lei;38;Switzerland;Passenger;no  
20;Khan Babar;39;Bulgaria;Passenger;yes  
./passenger_data.csv  
./passenger_data.csv  
less passenger_data.csv
```

## [5] “Στατιστικά-reports”

Αντίστοιχα με πριν έχουμε 4 χρήσεις τις awk για εξαγωγή στατιστικών μέσα από το csv μας.

Αρχικά δημιουργούμε ηλικιακές ομάδες ίδιες με αυτές που αναφέρεται στην εκφώνηση και όταν μια γραμμή έχει \$3 (ηλικία) μέσα σε αυτή ο

δείκτης της αυξάνεται κατά 1 και στο τέλος εμφανίζονται όλες και πόσοι “πέφτουν” στην καθεμία. Μετά για αυτές τις ηλικιακές ομάδες με τον ίδιο τρόπο βγάζουμε το ποσοστό όσων “διασώθηκαν” εξού και ο έλεγχος των rescued=Yes ή yes.

Ακολουθούν οι μέσοι όροι ηλικίας για τους απλούς επιβάτες αλλά και το προσωπικό με την ίδια λογική όπως πριν, άθροισμα ηλικίας επιβατών/προσωπικού διά το πλήθος τους, απλή εξαγωγή μέσου όρου. Τέλος πάλι με χρήση του print \$0 εκτυπώνονται όλες οι γραμμές/εγγραφές που διασώθηκαν, άρα φιλτράρουμε και παίρνουμε μόνο όσους έχουν rescued(\$6)=Yes|yes δηλ. μόνο τους διασωθέντες.

```
generate_reports(){
    input="$1"

    if [ -z "$input" ]; then
        input="./passenger_data.csv"
    elif [ ! -f "$input" ]; then
        echo "File does not exist."
        return
    fi

    # gia na ginontai rewrite kathe run
    > "ages.txt"
    > "percentages.txt"
    > "avg.txt"
    > "rescued.txt"

    awk -F'[;,]' '{
        age = $3;

        if (age >= 0 && age <= 18) {
            agegroup = "0-18";
        } else if (age >= 19 && age <= 35) {
            agegroup = "19-35";
        } else if (age >= 36 && age <= 50) {
            agegroup = "36-50";
        } else if (age >= 51) {
            agegroup = "51+";
        }

        ages[agegroup]++;
    } END {

        for (agegroup in ages) {
            print agegroup ":" ages[agegroup] " passengers";
        }
    }
```



```

}' "$input" > ages.txt

awk -F'[;,]' '{
    age = $3;

    if (age >= 0 && age <= 18) {
        agegroup = "0-18";
    } else if (age >= 19 && age <= 35) {
        agegroup = "19-35";
    } else if (age >= 36 && age <= 50) {
        agegroup = "36-50";
    } else if (age >= 51) {
        agegroup = "51+";
    }

    if ( $6 ~ /^[[:space:]]*(Yes|yes)[[:space:]]*$/ ) {
        rescued[agegroup]++;
    }
    total[agegroup]++;
} END {

    for (agegroup in total) {
        if (total[agegroup] > 0) {
            rescpercentage = int((rescued[agegroup] / total[agegroup]) * 100);
            print agegroup ":" rescued[agegroup] "/" total[agegroup] " rescued percentage: "rescpercentage"%";
        } else {
            print agegroup ": No data available for this agegroup";
        }
    }
}' "$input" > percentages.txt

awk -F'[;,]' '{
    age = $3;
    crew_or_pass = $5;

    if (age ~ /^[0-9]+([.][0-9]+)?$/ ) {
        total[crew_or_pass] += age;
        count[crew_or_pass]++;
    }
} END {
    for (crew_or_pass in total) {
        if (count[crew_or_pass] > 0) {
            average_age = total[crew_or_pass] / count[crew_or_pass];
            print crew_or_pass "s average age is: " average_age;
        }
    }
}
```

```

    }
}
}' "$input" > avg.txt

awk -F'[;,]' '{

    if ($6 ~ /^[[:space:]]*(Yes|yes)[[:space:]]*$/) {
        print $0
    }
}' "$input" > rescued.txt

echo "Generated:"
echo "Age groups in ages.txt"
echo "Rescue Percentages in percentages.txt"
echo "Average Age per Category in avg.txt"
echo "Rescued Passengers in rescued.txt"
}

```

## Παράδειγμα χρήσης:

The screenshot shows a VS Code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'ASKHSH1' with files: 'ages.txt', 'avg.txt', 'passenger\_data.csv', 'passengers\_ipc.sh', 'percentages.txt', 'processes\_ipc.sh', and 'rescued.txt'. The code editor shows the contents of 'processes\_ipc.sh', which is a shell script. The script includes a function 'insert\_data' and a main loop that processes data from 'passenger\_data.csv' and generates reports. The terminal at the bottom shows the command 'soktas@matebook:~/Documents/UniDocuments/Third Year/Operating Systems/CEID-OS-Projects/Project1/askhsh1\$ ./processes\_ipc.sh reports' and its output, which matches the 'Generated:' section of the script.

```

219 }' "$input" > percentages.txt
231 } END {
232     for (crew or pass in total) {
233         print $0
234     }
235 }
236 }' "$input" > avg.txt
237 }' "$input" > rescued.txt
238 }' "$input" > rescued.txt
239 }' "$input" > rescued.txt
240 }' "$input" > rescued.txt
241 }' "$input" > rescued.txt
242 }' "$input" > rescued.txt
243 }' "$input" > rescued.txt
244 }' "$input" > rescued.txt
245 }' "$input" > rescued.txt
246 }' "$input" > rescued.txt
247 }' "$input" > rescued.txt
248 }' "$input" > rescued.txt
249 }' "$input" > rescued.txt
250 }' "$input" > rescued.txt
251 }' "$input" > rescued.txt
252 }' "$input" > rescued.txt
253 }' "$input" > rescued.txt
254 }' "$input" > rescued.txt
255 }' "$input" > rescued.txt
256 }' "$input" > rescued.txt
257 }' "$input" > rescued.txt
258 }' "$input" > rescued.txt
259 }' "$input" > rescued.txt
260 }' "$input" > rescued.txt
261 if [ $1 == "insert" ]; then
262     insert_data
263 elif [ $1 == "reports" ]; then
264     reports
265 fi

```

Generated:  
Age groups in ages.txt  
Rescue Percentages in percentages.txt  
Average Age per Category in avg.txt  
Rescued Passengers in rescued.txt

```
processes_ipc.sh M  ages.txt M X  avg.txt M  percentages.txt M  rescued.txt M
ages.txt
You, 30 seconds ago | 1 author (You)
1 51+:589 passengers You, 30 seconds ago • Uncommitted changes
2 36-50:260 passengers
3 0-18:144 passengers
4 19-35:290 passengers
5
```

```
processes_ipc.sh M  ages.txt M  avg.txt M X  percentages.txt M  rescued.txt M
avg.txt
You, 45 seconds ago | 1 author (You)
1 Crews average age is: 46.5043 You, 45 seconds ago • Uncommitted changes
2 Passengers average age is: 47.5339
3
```

```
processes_ipc.sh M  ages.txt M  avg.txt M  percentages.txt M X  rescued.txt M
percentages.txt
You, 49 seconds ago | 1 author (You)
1 51+:484/589 rescued percentage: 82% You, 48 seconds ago • Uncommitted change
2 36-50:217/260 rescued percentage: 83%
3 0-18:122/144 rescued percentage: 84%
4 19-35:238/290 rescued percentage: 82%
5
```

```
processes_ipc.sh M  ages.txt M  avg.txt M  percentages.txt M  rescued.txt M X
rescued.txt > data
You, 52 seconds ago | 1 author (You)
1 1;Nunez Jorge;20;Russia;Passenger;Yes You, 4 weeks ago • Working askhsh 2 m
2 2;Hartmann Wolfgang;21;Germany;Passenger;Yes
3 3;Haarhoff Lily Tembo;22;United Kingdom;Passenger;yes
4 4;Zhang Wei;23;France;Passenger;yes
5 5;Zhang Yang;24;Italy;Passenger;yes
6 6;Nguyen Cam;25;Spain;Passenger;yes
7 7;Znaimer Moses;26;Poland;Passenger;yes
```

## Ασκηση 2

Η άσκηση 2 έχει υλοποιηθεί όπως λέει η εκφώνηση σε 3 αρχεία, 2 C και ένα header file (και το Makefile). Με κατάλληλη χρήση 3 σημαφόρων, των counting σημαφόρων `boat_space` και `boats_available` με τιμές ίσες με τις συνολικές θέσεις κάθε βάρκας και τις συνολικές βάρκες αντίστοιχα, για τους οποίους κάνουμε `wait (-1)` μέχρι να γεμίσουν οι θέσεις στη βάρκα, για κάθε βάρκα (-1 την τιμή του αν χωράει άλλος επιβάτης και όταν γεμίσει κάνει `wait` τον σημαφόρο των συνολικών λέμβων συμβολίζοντας ότι γέμισε η μια από αυτές), και του binary σημαφόρου `boarding_queue` για να εκτελεί ο κάθε επιβάτης την επιβίβαση του χωρίς την παρέμβαση των άλλων (αμοιβαίος αποκλεισμός), καταφέρνουμε να συγχρονίσουμε την δρομολόγηση λέμβων για την προσομοίωση διάσωσης. Η λογική θα εξηγηθεί και άλλο παρακάτω.

Εδώ είναι το header file μας `ipc_utils.h` για την δήλωση των σημαφόρων και των βασικών συναρτήσεων:

```
#include <semaphore.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <time.h>
#include <unistd.h> //για th sleep

// global shmaforoi
extern sem_t boat_space;
extern sem_t boarding_queue;
extern sem_t boats_available;
```

```
void init_semaphore(sem_t *sem, int value);
void destroy_semaphore(sem_t *sem);
void wait_semaphore(sem_t *sem);
void post_semaphore(sem_t *sem);

void passenger_routine(int passenger_id, int total_passengers, int
totalboats, int seats_per_boat);
```

Έπειτα είναι το αρχείο passenger.c το οποίο κάνει include το header file με όλες τις απαραίτητες βιβλιοθήκες αλλά και τα function declarations αυτών που θα χρειαστεί. Αρχικά οφείλουμε να πούμε ότι δημιουργήσαμε κάποιες μικρές συναρτήσεις για error handling των συναρτήσεων των σηματοφόρων. Αυτές δεν κάνουν τίποτα άλλο παρά την ίδια την διεργασία που κάνουν οι συναρτήσεις της βιβλιοθήκης semaphore.h απλά αν προκύψει error το εκτυπώνει και τελειώνει το πρόγραμμα.

Η ουσία του passenger.c είναι η συνάρτηση passenger\_routine. Σε αυτή γίνονται τα παραπάνω που αναφέραμε. Απομονώνοντας κάθε επιβίβαση επιβάτη μέσα στο wait\_semaphore(&boarding\_queue) ..... post\_semaphore(&boarding\_queue) κάνουμε σίγουρο το ότι κάθε επιβάτης δεν θα επηρεαστεί από τους επόμενους γιατί ο σηματοφόρος είναι binary άρα περιμένουν (wait) οι άλλοι να τελειώσει(post).

Κάθε επιβάτης λοιπόν κάνει wait και δεσμεύει μια θέση στη βάρκα μέχρι να γίνει η τιμή του σημαφόρου boat\_space = 0 και άρα passengers\_in\_boat == seats\_per\_boat. Τότε δεσμεύεται μια βάρκα (wait) και ελευθερώνεται ο σημαφόρος boat\_space κάνοντας όσα post όσοι και οι επιβάτες/θέσεις σε αυτή. Όταν έχουν δεσμευτεί όλες οι βάρκες, δηλ ο σημαφόρος boats\_available = 0 και boats\_filled == totalboats τότε έρχεται η ώρα για την αναχώρηση των λέμβων. Αποδεσμεύονται όλες κάνοντας post τον σημαφόρο boats\_available τόσες φορές όσες δεσμεύτηκε(totalboats) κάτι που θα σημάνει την αναχώρηση των λεμβών και μετά από 2 δευτερόλεπτα επιστρέφουν και πάλι από την αρχή μέχρι να διασωθούν όλοι!

```
#include "ipc_utils.h"

extern sem_t boat_space;
extern sem_t boarding_queue;
extern sem_t boats_available;

void init_semaphore(sem_t *sem, int value) {
    if (sem_init(sem, 0, value) != 0) {
        perror("sem_init failed");
        exit(1);
    }
}

void destroy_semaphore(sem_t *sem) {
    if (sem_destroy(sem) != 0) {
        perror("sem_destroy failed");
        exit(1);
    }
}

void wait_semaphore(sem_t *sem) {
    if (sem_wait(sem) != 0) {
```

```

        perror("sem_wait failed");
        exit(1);
    }
}

void post_semaphore(sem_t *sem) {
    if (sem_post(sem) != 0) {
        perror("sem_post failed");
        exit(1);
    }
}

void passenger_routine(int passenger_id, int total_passengers, int totalboats,
int seats_per_boat) {
    static int remaining_passengers = -1;
    static int passengers_in_boat = 0;
    static int boats_filled = 0;

    if (remaining_passengers == -1) {
        remaining_passengers = total_passengers;
    }

    if (remaining_passengers <= 0) {
        return;
    }

    wait_semaphore(&boarding_queue);

    if (remaining_passengers > 0) {
        wait_semaphore(&boat_space);

        remaining_passengers--;
        printf("Passenger %d boards the boat no %d. Remaining passengers: %d\n",
passenger_id, boats_filled+1 ,remaining_passengers);

        passengers_in_boat++;

        if (passengers_in_boat == seats_per_boat) {

            wait_semaphore(&boats_available);
            boats_filled++;

            if (boats_filled == totalboats) {
                printf("%d boat(s) depart(s) with %d passengers each.\n",

```

```

boats_filled, passengers_in_boat);
    sleep(2);
    printf("The boats have returned.\n");
    boats_filled = 0;
    for (int i = 0; i < totalboats; i++) {
        post_semaphore(&boats_available);
    }
}
for (int i = 0; i < passengers_in_boat; i++) {
    post_semaphore(&boat_space);
}
passengers_in_boat = 0;
}
}

post_semaphore(&boarding_queue);
}

```

Τέλος έχουμε το βασικό αρχείο `launch.c`, την “main” μας, η οποία αρχικοποιεί τις τιμές και κάνει handle το queue με τους επιβάτες. Έχουμε ένα μέγιστο όριο 200 επιβατών και 50 λέμβων που μπορεί κάλλιστα να αλλάξει σε ότι αριθμούς επιθυμούμε. Αφού πάρουμε συνολικούς επιβάτες, βάρκες και θέσεις σε κάθε βάρκα, φτιάχνουμε την ουρά και με μια πιθανότητα 20% να αλλάξει ο επιβάτης γνώμη, τον κάνουμε `reqqueue shift`-άροντας την ουρά μια θέση αριστερά και θέτοντας την τελευταία τιμή ίση με το `id` του. Αν όμως δεν αλλάξει κάνουμε το αντίστοιχο `shift` αλλά μειώνουμε το μέγεθος της ουράς κατά 1 καθώς πλέον επιβιβάστηκε.

```

#include "ipc_utils.h"

#define MAX_PASSENGERS 200
#define MAX_BOATS 50

```



```

sem_t boat_space;
sem_t boarding_queue;
sem_t boats_available;

int main() {

    srand(time(NULL));

    int passengers, boats, seats_per_boat;

    printf("Enter number of passengers: ");
    scanf("%d", &passengers);
    printf("Enter number of boats: ");
    scanf("%d", &boats);
    printf("Enter seats per boat: ");
    scanf("%d", &seats_per_boat);

    if (passengers > MAX_PASSENGERS || boats > MAX_BOATS || seats_per_boat
<= 0) {
        printf("Invalid input! Please check the limits.\n");
        return 1;
    }

    int queue[passengers];
    int queue_size = passengers;

    // Arxikopoihsh ouras epibatwn
    for (int i = 0; i < passengers; i++) {
        queue[i] = i + 1;
    }

    // Arxikopoihsh semaphorwn
    init_semaphore(&boat_space, seats_per_boat);
    init_semaphore(&boarding_queue, 1);
    init_semaphore(&boats_available, boats);

    while (queue_size > 0) {
        int passenger_id = queue[0];

        // 20% pithanothta na metakinithei o epibaths sto telos ths ouras
        if ((rand() % 100) < 20) {

```

```

        printf("Passenger %d chooses not to board and moves to the back of
the queue.\n", passenger_id);

        // an den epibastei kanoume shift thn oura
        for (int j = 0; j < queue_size - 1; j++) {
            queue[j] = queue[j + 1];
        }

        //kai paei sto telos
        queue[queue_size - 1] = passenger_id;

    } else {
        // Kalesma synarthshs gia thn diadikasia epibivashs
        passenger_routine(passenger_id, passengers, boats, seats_per_boat);

        //kathe fora pairnw to passenger_id ws queue[0] ara to front, kai
afou phra to front shiftarw gia na paei o epomenos
        for (int j = 0; j < queue_size - 1; j++) {
            queue[j] = queue[j + 1];
        }

        queue_size--;
    }
}

destroy_semaphore(&boat_space);
destroy_semaphore(&boarding_queue);
destroy_semaphore(&boats_available);

printf("All passengers have been rescued!\n");
return 0;
}

```

## Παραδείγματα εκτέλεσης:

- Ίσος αριθμός βαρκών και θέσεων:

```
• soktas@matebook:~/Documents/UniDocuments/Third Year/Operating Systems/CEID-OS-Projects/Project1/askhsh2$ make
gcc launch.c passenger.c -g -o output/launch -Wall -Werror -Wextra -lm && ./output/launch
Enter number of passengers: 25
Enter number of boats: 2
Enter seats per boat: 2
Passenger 1 boards the boat no 1. Remaining passengers: 24
Passenger 2 chooses not to board and moves to the back of the queue.
Passenger 3 boards the boat no 1. Remaining passengers: 23
Passenger 4 boards the boat no 2. Remaining passengers: 22
Passenger 5 boards the boat no 2. Remaining passengers: 21
2 boat(s) depart(s) with 2 passengers each.
The boats have returned.
Passenger 6 chooses not to board and moves to the back of the queue.
Passenger 7 boards the boat no 1. Remaining passengers: 20
Passenger 8 boards the boat no 1. Remaining passengers: 19
Passenger 9 boards the boat no 2. Remaining passengers: 18
Passenger 10 boards the boat no 2. Remaining passengers: 17
2 boat(s) depart(s) with 2 passengers each.
The boats have returned.
Passenger 11 boards the boat no 1. Remaining passengers: 16
Passenger 12 chooses not to board and moves to the back of the queue.
Passenger 13 chooses not to board and moves to the back of the queue.
Passenger 14 boards the boat no 1. Remaining passengers: 15
Passenger 15 chooses not to board and moves to the back of the queue.
Passenger 16 boards the boat no 2. Remaining passengers: 14
Passenger 17 boards the boat no 2. Remaining passengers: 13
2 boat(s) depart(s) with 2 passengers each.
The boats have returned.
Passenger 18 boards the boat no 1. Remaining passengers: 12
Passenger 19 boards the boat no 1. Remaining passengers: 11
Passenger 20 boards the boat no 2. Remaining passengers: 10
Passenger 21 boards the boat no 2. Remaining passengers: 9
2 boat(s) depart(s) with 2 passengers each.
The boats have returned.
Passenger 22 boards the boat no 1. Remaining passengers: 8
Passenger 23 boards the boat no 1. Remaining passengers: 7
Passenger 24 boards the boat no 2. Remaining passengers: 6
Passenger 25 chooses not to board and moves to the back of the queue.
Passenger 2 boards the boat no 2. Remaining passengers: 5
2 boat(s) depart(s) with 2 passengers each.
The boats have returned.
Passenger 6 boards the boat no 1. Remaining passengers: 4
Passenger 12 boards the boat no 1. Remaining passengers: 3
Passenger 13 boards the boat no 2. Remaining passengers: 2
Passenger 15 boards the boat no 2. Remaining passengers: 1
2 boat(s) depart(s) with 2 passengers each.
The boats have returned.
Passenger 25 boards the boat no 1. Remaining passengers: 0
All passengers have been rescued!
```

- Αριθμός βαρκών > αριθμό θέσεων:

```
• soktas@matebook:~/Documents/UniDocuments/Third Year/Operating Systems/CEID-OS-Projects/Project1/askhsh2$ make
gcc launch.c passenger.c -g -o output/launch -Wall -Werror -Wextra -lm && ./output/launch
Enter number of passengers: 32
Enter number of boats: 5
Enter seats per boat: 3
Passenger 1 boards the boat no 1. Remaining passengers: 31
Passenger 2 chooses not to board and moves to the back of the queue.
Passenger 3 chooses not to board and moves to the back of the queue.
Passenger 4 boards the boat no 1. Remaining passengers: 30
Passenger 5 boards the boat no 1. Remaining passengers: 29
Passenger 6 boards the boat no 2. Remaining passengers: 28
Passenger 7 boards the boat no 2. Remaining passengers: 27
Passenger 8 boards the boat no 2. Remaining passengers: 26
Passenger 9 chooses not to board and moves to the back of the queue.
Passenger 10 boards the boat no 3. Remaining passengers: 25
Passenger 11 boards the boat no 3. Remaining passengers: 24
Passenger 12 boards the boat no 3. Remaining passengers: 23
Passenger 13 boards the boat no 4. Remaining passengers: 22
Passenger 14 boards the boat no 4. Remaining passengers: 21
Passenger 15 boards the boat no 4. Remaining passengers: 20
Passenger 16 chooses not to board and moves to the back of the queue.
Passenger 17 boards the boat no 5. Remaining passengers: 19
Passenger 18 boards the boat no 5. Remaining passengers: 18
Passenger 19 boards the boat no 5. Remaining passengers: 17
5 boat(s) depart(s) with 3 passengers each.
The boats have returned.
Passenger 20 boards the boat no 1. Remaining passengers: 16
Passenger 21 boards the boat no 1. Remaining passengers: 15
Passenger 22 boards the boat no 1. Remaining passengers: 14
Passenger 23 boards the boat no 2. Remaining passengers: 13
Passenger 24 boards the boat no 2. Remaining passengers: 12
Passenger 25 boards the boat no 2. Remaining passengers: 11
Passenger 26 boards the boat no 3. Remaining passengers: 10
Passenger 27 boards the boat no 3. Remaining passengers: 9
Passenger 28 boards the boat no 3. Remaining passengers: 8
Passenger 29 boards the boat no 4. Remaining passengers: 7
Passenger 30 boards the boat no 4. Remaining passengers: 6
Passenger 31 chooses not to board and moves to the back of the queue.
Passenger 32 boards the boat no 4. Remaining passengers: 5
Passenger 2 boards the boat no 5. Remaining passengers: 4
Passenger 3 boards the boat no 5. Remaining passengers: 3
Passenger 9 boards the boat no 5. Remaining passengers: 2
5 boat(s) depart(s) with 3 passengers each.
The boats have returned.
Passenger 16 boards the boat no 1. Remaining passengers: 1
Passenger 31 boards the boat no 1. Remaining passengers: 0
All passengers have been rescued!
```

- Αριθμός βαρκών < αριθμό θέσεων:

```

soktas@matebook:~/Documents/UniDocuments/Third Year/Operating Systems/CEID-OS-Projects/Project1/askhsh2$ make
gcc launch.c passenger.c -g -o output/launch -Wall -Werror -Wextra -lm && ./output/launch
Enter number of passengers: 12
Enter number of boats: 2
Enter seats per boat: 4
Passenger 1 boards the boat no 1. Remaining passengers: 11
Passenger 2 boards the boat no 1. Remaining passengers: 10
Passenger 3 boards the boat no 1. Remaining passengers: 9
Passenger 4 boards the boat no 1. Remaining passengers: 8
Passenger 5 boards the boat no 2. Remaining passengers: 7
Passenger 6 boards the boat no 2. Remaining passengers: 6
Passenger 7 boards the boat no 2. Remaining passengers: 5
Passenger 8 boards the boat no 2. Remaining passengers: 4
2 boat(s) depart(s) with 4 passengers each.
The boats have returned.
Passenger 9 boards the boat no 1. Remaining passengers: 3
Passenger 10 boards the boat no 1. Remaining passengers: 2
Passenger 11 chooses not to board and moves to the back of the queue.
Passenger 12 boards the boat no 1. Remaining passengers: 1
Passenger 11 boards the boat no 1. Remaining passengers: 0
All passengers have been rescued!

```

- Εκτέλεση με μεγάλο αριθμό επιβατών:

```

soktas@matebook:~/Documents/UniDocuments/Third Year/Operating Systems/CEID-OS-Projects/Project1/askhsh2$ make
gcc launch.c passenger.c -g -o output/launch -Wall -Werror -Wextra -lm && ./output/launch
Enter number of passengers: 157
Enter number of boats: 12
Enter seats per boat: 5
Passenger 1 boards the boat no 1. Remaining passengers: 156
Passenger 2 chooses not to board and moves to the back of the queue.
Passenger 3 boards the boat no 1. Remaining passengers: 155
Passenger 4 chooses not to board and moves to the back of the queue.
Passenger 5 boards the boat no 1. Remaining passengers: 154
Passenger 6 boards the boat no 1. Remaining passengers: 153
Passenger 7 boards the boat no 1. Remaining passengers: 152
Passenger 8 boards the boat no 2. Remaining passengers: 151
Passenger 9 boards the boat no 2. Remaining passengers: 150
Passenger 10 boards the boat no 2. Remaining passengers: 149
Passenger 11 boards the boat no 2. Remaining passengers: 148
Passenger 12 boards the boat no 2. Remaining passengers: 147
Passenger 13 boards the boat no 3. Remaining passengers: 146
Passenger 14 boards the boat no 3. Remaining passengers: 145
Passenger 15 chooses not to board and moves to the back of the queue.
Passenger 16 boards the boat no 3. Remaining passengers: 144
Passenger 17 chooses not to board and moves to the back of the queue.
Passenger 18 boards the boat no 3. Remaining passengers: 143
Passenger 19 chooses not to board and moves to the back of the queue.
Passenger 20 chooses not to board and moves to the back of the queue.
Passenger 21 boards the boat no 3. Remaining passengers: 142
Passenger 22 boards the boat no 4. Remaining passengers: 141
Passenger 23 boards the boat no 4. Remaining passengers: 140
Passenger 24 boards the boat no 4. Remaining passengers: 139
Passenger 25 boards the boat no 4. Remaining passengers: 138
Passenger 26 boards the boat no 4. Remaining passengers: 137
Passenger 27 chooses not to board and moves to the back of the queue.
Passenger 28 boards the boat no 5. Remaining passengers: 136
Passenger 29 boards the boat no 5. Remaining passengers: 135
Passenger 30 boards the boat no 5. Remaining passengers: 134
Passenger 31 boards the boat no 5. Remaining passengers: 133
Passenger 32 boards the boat no 5. Remaining passengers: 132
Passenger 33 chooses not to board and moves to the back of the queue.
Passenger 34 chooses not to board and moves to the back of the queue.
Passenger 35 boards the boat no 6. Remaining passengers: 131
Passenger 36 chooses not to board and moves to the back of the queue.
Passenger 37 boards the boat no 6. Remaining passengers: 130
Passenger 38 boards the boat no 6. Remaining passengers: 129
Passenger 39 boards the boat no 6. Remaining passengers: 128
Passenger 40 boards the boat no 6. Remaining passengers: 127
Passenger 41 boards the boat no 7. Remaining passengers: 126
Passenger 42 chooses not to board and moves to the back of the queue.
Passenger 43 boards the boat no 7. Remaining passengers: 125
Passenger 44 boards the boat no 7. Remaining passengers: 124
Passenger 45 chooses not to board and moves to the back of the queue.
Passenger 46 chooses not to board and moves to the back of the queue.
Passenger 47 chooses not to board and moves to the back of the queue.
Passenger 48 boards the boat no 7. Remaining passengers: 123

```

Passenger 49 boards the boat no 7. Remaining passengers: 122  
Passenger 50 chooses not to board and moves to the back of the queue.  
Passenger 51 boards the boat no 8. Remaining passengers: 121  
Passenger 52 boards the boat no 8. Remaining passengers: 120  
Passenger 53 boards the boat no 8. Remaining passengers: 119  
Passenger 54 chooses not to board and moves to the back of the queue.  
Passenger 55 boards the boat no 8. Remaining passengers: 118  
Passenger 56 boards the boat no 8. Remaining passengers: 117  
Passenger 57 boards the boat no 9. Remaining passengers: 116  
Passenger 58 boards the boat no 9. Remaining passengers: 115  
Passenger 59 boards the boat no 9. Remaining passengers: 114  
Passenger 60 boards the boat no 9. Remaining passengers: 113  
Passenger 61 boards the boat no 9. Remaining passengers: 112  
Passenger 62 boards the boat no 10. Remaining passengers: 111  
Passenger 63 boards the boat no 10. Remaining passengers: 110  
Passenger 64 boards the boat no 10. Remaining passengers: 109  
Passenger 65 chooses not to board and moves to the back of the queue.  
Passenger 66 boards the boat no 10. Remaining passengers: 108  
Passenger 67 boards the boat no 10. Remaining passengers: 107  
Passenger 68 boards the boat no 11. Remaining passengers: 106  
Passenger 69 boards the boat no 11. Remaining passengers: 105  
Passenger 70 boards the boat no 11. Remaining passengers: 104  
Passenger 71 boards the boat no 11. Remaining passengers: 103  
Passenger 72 boards the boat no 11. Remaining passengers: 102  
Passenger 73 boards the boat no 12. Remaining passengers: 101  
Passenger 74 boards the boat no 12. Remaining passengers: 100  
Passenger 75 chooses not to board and moves to the back of the queue.  
Passenger 76 boards the boat no 12. Remaining passengers: 99  
Passenger 77 boards the boat no 12. Remaining passengers: 98  
Passenger 78 boards the boat no 12. Remaining passengers: 97  
12 boat(s) depart(s) with 5 passengers each.  
The boats have returned.  
Passenger 79 boards the boat no 1. Remaining passengers: 96  
Passenger 80 chooses not to board and moves to the back of the queue.  
Passenger 81 boards the boat no 1. Remaining passengers: 95  
Passenger 82 boards the boat no 1. Remaining passengers: 94  
Passenger 83 boards the boat no 1. Remaining passengers: 93  
Passenger 84 boards the boat no 1. Remaining passengers: 92  
Passenger 85 boards the boat no 2. Remaining passengers: 91  
Passenger 86 chooses not to board and moves to the back of the queue.  
Passenger 87 boards the boat no 2. Remaining passengers: 90  
Passenger 88 boards the boat no 2. Remaining passengers: 89  
Passenger 89 boards the boat no 2. Remaining passengers: 88  
Passenger 90 chooses not to board and moves to the back of the queue.  
Passenger 91 boards the boat no 2. Remaining passengers: 87  
Passenger 92 boards the boat no 3. Remaining passengers: 86  
Passenger 93 chooses not to board and moves to the back of the queue.  
Passenger 94 boards the boat no 3. Remaining passengers: 85  
Passenger 95 boards the boat no 3. Remaining passengers: 84  
Passenger 96 boards the boat no 3. Remaining passengers: 83  
Passenger 97 boards the boat no 3. Remaining passengers: 82  
Passenger 98 boards the boat no 4. Remaining passengers: 81  
Passenger 99 boards the boat no 4. Remaining passengers: 80  
Passenger 100 chooses not to board and moves to the back of the queue.  
Passenger 101 chooses not to board and moves to the back of the queue.  
Passenger 102 boards the boat no 4. Remaining passengers: 79  
Passenger 103 boards the boat no 4. Remaining passengers: 78  
Passenger 104 boards the boat no 4. Remaining passengers: 77  
Passenger 105 boards the boat no 5. Remaining passengers: 76  
Passenger 106 boards the boat no 5. Remaining passengers: 75  
Passenger 107 boards the boat no 5. Remaining passengers: 74  
Passenger 108 boards the boat no 5. Remaining passengers: 73

Passenger 108 boards the boat no 5. Remaining passengers: 73  
Passenger 109 chooses not to board and moves to the back of the queue.  
Passenger 110 boards the boat no 5. Remaining passengers: 72  
Passenger 111 boards the boat no 6. Remaining passengers: 71  
Passenger 112 boards the boat no 6. Remaining passengers: 70  
Passenger 113 boards the boat no 6. Remaining passengers: 69  
Passenger 114 boards the boat no 6. Remaining passengers: 68  
Passenger 115 chooses not to board and moves to the back of the queue.  
Passenger 116 boards the boat no 6. Remaining passengers: 67  
Passenger 117 chooses not to board and moves to the back of the queue.  
Passenger 118 chooses not to board and moves to the back of the queue.  
Passenger 119 boards the boat no 7. Remaining passengers: 66  
Passenger 120 boards the boat no 7. Remaining passengers: 65  
Passenger 121 boards the boat no 7. Remaining passengers: 64  
Passenger 122 chooses not to board and moves to the back of the queue.  
Passenger 123 boards the boat no 7. Remaining passengers: 63  
Passenger 124 boards the boat no 7. Remaining passengers: 62  
Passenger 125 boards the boat no 8. Remaining passengers: 61  
Passenger 126 boards the boat no 8. Remaining passengers: 60  
Passenger 127 boards the boat no 8. Remaining passengers: 59  
Passenger 128 chooses not to board and moves to the back of the queue.  
Passenger 129 boards the boat no 8. Remaining passengers: 58  
Passenger 130 boards the boat no 8. Remaining passengers: 57  
Passenger 131 boards the boat no 9. Remaining passengers: 56  
Passenger 132 chooses not to board and moves to the back of the queue.  
Passenger 133 boards the boat no 9. Remaining passengers: 55  
Passenger 134 chooses not to board and moves to the back of the queue.  
Passenger 135 boards the boat no 9. Remaining passengers: 54  
Passenger 136 boards the boat no 9. Remaining passengers: 53  
Passenger 137 boards the boat no 9. Remaining passengers: 52  
Passenger 138 boards the boat no 10. Remaining passengers: 51  
Passenger 139 chooses not to board and moves to the back of the queue.  
Passenger 140 boards the boat no 10. Remaining passengers: 50  
Passenger 141 chooses not to board and moves to the back of the queue.  
Passenger 142 boards the boat no 10. Remaining passengers: 49  
Passenger 143 boards the boat no 10. Remaining passengers: 48  
Passenger 144 boards the boat no 10. Remaining passengers: 47  
Passenger 145 boards the boat no 11. Remaining passengers: 46  
Passenger 146 boards the boat no 11. Remaining passengers: 45  
Passenger 147 boards the boat no 11. Remaining passengers: 44  
Passenger 148 chooses not to board and moves to the back of the queue.  
Passenger 149 boards the boat no 11. Remaining passengers: 43  
Passenger 150 boards the boat no 11. Remaining passengers: 42  
Passenger 151 boards the boat no 12. Remaining passengers: 41  
Passenger 152 boards the boat no 12. Remaining passengers: 40  
Passenger 153 boards the boat no 12. Remaining passengers: 39  
Passenger 154 boards the boat no 12. Remaining passengers: 38  
Passenger 155 boards the boat no 12. Remaining passengers: 37  
12 boat(s) depart(s) with 5 passengers each.  
The boats have returned.  
Passenger 156 boards the boat no 1. Remaining passengers: 36  
Passenger 157 chooses not to board and moves to the back of the queue.  
Passenger 2 boards the boat no 1. Remaining passengers: 35  
Passenger 4 boards the boat no 1. Remaining passengers: 34  
Passenger 15 boards the boat no 1. Remaining passengers: 33  
Passenger 17 boards the boat no 1. Remaining passengers: 32  
Passenger 19 boards the boat no 2. Remaining passengers: 31  
Passenger 20 boards the boat no 2. Remaining passengers: 30  
Passenger 27 chooses not to board and moves to the back of the queue.  
Passenger 33 chooses not to board and moves to the back of the queue.  
Passenger 34 chooses not to board and moves to the back of the queue.  
Passenger 36 boards the boat no 2. Remaining passengers: 29



```
Passenger 34 chooses not to board and moves to the back of the queue.
Passenger 36 boards the boat no 2. Remaining passengers: 29
Passenger 42 boards the boat no 2. Remaining passengers: 28
Passenger 45 chooses not to board and moves to the back of the queue.
Passenger 46 chooses not to board and moves to the back of the queue.
Passenger 47 boards the boat no 2. Remaining passengers: 27
Passenger 50 boards the boat no 3. Remaining passengers: 26
Passenger 54 boards the boat no 3. Remaining passengers: 25
Passenger 65 boards the boat no 3. Remaining passengers: 24
Passenger 75 boards the boat no 3. Remaining passengers: 23
Passenger 80 boards the boat no 3. Remaining passengers: 22
Passenger 86 boards the boat no 4. Remaining passengers: 21
Passenger 90 boards the boat no 4. Remaining passengers: 20
Passenger 93 boards the boat no 4. Remaining passengers: 19
Passenger 100 boards the boat no 4. Remaining passengers: 18
Passenger 101 boards the boat no 4. Remaining passengers: 17
Passenger 109 boards the boat no 5. Remaining passengers: 16
Passenger 115 chooses not to board and moves to the back of the queue.
Passenger 117 chooses not to board and moves to the back of the queue.
Passenger 118 boards the boat no 5. Remaining passengers: 15
Passenger 122 boards the boat no 5. Remaining passengers: 14
Passenger 128 boards the boat no 5. Remaining passengers: 13
Passenger 132 boards the boat no 5. Remaining passengers: 12
Passenger 134 boards the boat no 6. Remaining passengers: 11
Passenger 139 boards the boat no 6. Remaining passengers: 10
Passenger 141 boards the boat no 6. Remaining passengers: 9
Passenger 148 chooses not to board and moves to the back of the queue.
Passenger 157 boards the boat no 6. Remaining passengers: 8
Passenger 27 boards the boat no 6. Remaining passengers: 7
Passenger 33 boards the boat no 7. Remaining passengers: 6
Passenger 34 boards the boat no 7. Remaining passengers: 5
Passenger 45 chooses not to board and moves to the back of the queue.
Passenger 46 boards the boat no 7. Remaining passengers: 4
Passenger 115 chooses not to board and moves to the back of the queue.
Passenger 117 chooses not to board and moves to the back of the queue.
Passenger 148 boards the boat no 7. Remaining passengers: 3
Passenger 45 boards the boat no 7. Remaining passengers: 2
Passenger 115 boards the boat no 8. Remaining passengers: 1
Passenger 117 boards the boat no 8. Remaining passengers: 0
All passengers have been rescued!
```



### Ασκηση 3

Το παρακάτω πρόγραμμα υλοποιεί μια προσομοίωση χρονοδρομολόγησης διεργασιών και δυναμικής μνήμης μεγέθους 512KB. Κάθε διεργασία (process) περιγράφεται από ένα id (pid), τον χρόνο άφιξής της στο σύστημα, τον απαιτούμενο χρόνο εκτέλεσης, τον υπολειπόμενο χρόνο εκτέλεσης (remaining\_time), τη μνήμη που απαιτεί (req\_memory) και μια ένδειξη ύπαρξης στη μνήμη (in\_memory). Ο αλγόριθμος που χρησιμοποιείται για την χρονοδρομολόγηση είναι Round Robin με κβάντο 3ms, ενώ για την τοποθέτηση μιας διεργασίας στη μνήμη εφαρμόζεται first-fit.

```
#define MEMORY_SIZE 512 // megethos mnimis KB
#define TQ 3 // Round Robin time quantum ms

typedef struct {
    int pid;
    int arrival_time;
    int duration;
    int remaining_time;
    int req_memory;
    bool in_memory;
} Process;

typedef struct {
    int start;
    int size;
    bool isfree;
    int pid;
} Memory_Block;
```

Η μνήμη έχει μοντελοποιηθεί ως ένας πίνακας τύπου `Memory_Block` μεγέθους 512, όπου κάθε κελί του αναπαριστά ένα δυναμικό block με πεδίο `start`, `size`, `isfree` και `pid`. Αρχικά όλη η μνήμη (όπως αναφέρει και η εκφώνηση) είναι ελεύθερη (ένα block 512 KB).

```
Memory_Block memory[MEMORY_SIZE];  
// arxikopisi dynamic mnimis, xoris na eisaxthei kapoia diergasia akoma  
void memory_init(){  
    memory[0].start = 0;  
    memory[0].size = MEMORY_SIZE;  
    memory[0].isfree = true;  
    memory[0].pid = -1;    //because free  
    num_blocks = 1;  
}
```

Όταν καταφτάσει μια διεργασία με ορισμένο απαιτούμενο χώρο, ο αλγόριθμος αναζητά το πρώτο ελεύθερο μπλοκ που αρκεί για να την χωρέσει. Εάν η διεργασία χωράει, εισέρχεται στη μνήμη και δημιουργείται 2ο μπλοκ με το την υπολειπόμενη ελεύθερη μνήμη. Κατά την ολοκλήρωση μιας διεργασίας, το αντίστοιχο τμήμα της μνήμης απελευθερώνεται και (όταν μπορεί) συγχωνεύεται με διπλανά ελεύθερα μπλοκ (defragmentation).

```
bool allocate_memory(Process *p){  
    for(int i=0; i<num_blocks; ++i){  
        if(memory[i].isfree && memory[i].size >= p->req_memory){  
            if(memory[i].size > p->req_memory){  
                for(int j=num_blocks; j>i; j--) memory[j] = memory[j-1];  
                num_blocks++;  
                //to 2o miso toy new block pou einai empty (is_empty() == true)  
                memory[i+1].start = memory[i].start + p->req_memory;  
                memory[i+1].size=memory[i].size-p->req_memory; memory[i+1].isfree=true; memory[i+1].pid=-1;  
            }  
            //to 1o miso toy kainourioy block me to process  
            memory[i].size = p->req_memory;  
            memory[i].isfree = false; memory[i].pid = p->pid; p->in_memory = true;  
            return true;  
        }  
    }  
    return false;  
}
```

```

bool deallocate_memory(int pid){
    int k = -1;
    //find and deallocate mem block
    for (int i = 0; i < num_blocks; ++i) {
        if (memory[i].pid == pid) {
            memory[i].isfree = true;
            memory[i].pid = -1;
            k = i;
            break;
        }
    }
    // If the block with the given PID was not found
    if (k == -1) {
        return false;
    }
    // kane merge me previous block if free (de-fragmentation)
    if (k > 0 && memory[k - 1].isfree) {
        memory[k - 1].size += memory[k].size;
        for (int j = k; j < num_blocks - 1; ++j) {
            memory[j] = memory[j + 1];
        }
        num_blocks--;
        k--; //kanonikoposi epeidi meiothike kata 1 block i mnimi
    }
    //tsekarei to idio me to mprostino block
    if (k < num_blocks - 1 && memory[k + 1].isfree) {
        memory[k].size += memory[k + 1].size;
        for (int j = k + 1; j < num_blocks - 1; ++j) {
            memory[j] = memory[j + 1];
        }
        num_blocks--;
    }
    return true;
}

```

Όσον αφορά την χρονοδρομολόγηση, σε κάθε διεργασία δίνεται 3ms χρόνος εκτέλεσης κάθε φορά (ή λιγότερο αν η διεργασία απαιτεί μικρότερο χρόνο εκτέλεσης). Εάν μια εργασία δεν βρίσκεται στη μνήμη όταν έρθει η σειρά της και δεν υπάρχει διαθέσιμος χώρος τότε παρακάμπτεται (προσωρινά) μέχρι να ελευθερωθεί κάποιος αρκετός χώρος. Επιπλέον, για να ισχύει η έννοια του arrival\_time κάθε διεργασίας, τότε θα πρέπει να ελέγχεται μαζί με το αν δεν έχει ολοκληρωθεί και εάν έχει έρθει στο πρόγραμμα (arrival\_time <= time). Για να αντιμετωπιστεί αυτό δημιουργήσαμε μια συνάρτηση (next\_arrival\_time) η οποία επιστρέφει τον χρόνο της επόμενης διεργασίας για να προχωρήσει ο επεξεργαστής σε εκτέλεση.

```
void round_robin(Process processes[], int p){
    int time = 0;
    bool done = false;

    while(!done){
        done = true; //Assume all processes are completed
        bool any_ready = false; //an brikame kapoio process auton ton kiklo (arrival_time
related)

        for(int i = 0; i < p; ++i){
            if(processes[i].remaining_time > 0 && processes[i].arrival_time <= time){
//an den exei teleiosei KAI exei erthei.
                done = false; //Brethike process me != 0 burst time
                any_ready = true;

                //AN DEN einai stin mnimi, tote allocate
                if (!processes[i].in_memory) {
                    if(!allocate_memory(&processes[i])) continue; //skip process exec if
allocation fails
                }

                //burst time > time quantum
                else if(processes[i].remaining_time > TQ){
                    time += TQ;
                    processes[i].remaining_time -= TQ;
                }
                //Process CAN BE completed
            else{
                time += processes[i].remaining_time;
                processes[i].remaining_time = 0;
            }
        }
    }
}
```

```

        deallocate_memory(processes[i].pid);
        processes[i].in_memory = false; //deallocated
    }
    printf("\n[Time = %d] | Process %d is scheduled.\n", time,
processes[i].pid);
    print_memory();
    print_process_status(processes, p);
}
}
if(!any_ready){
    int jump = next_arrival_time(processes, p, time);
    if (jump == -1) break; //afou den vrike kati tote teleiosame
    else {
        time = jump; //skip forward in time gia an vrei to epomeno arriving process
        done = false;
    }
}
}
}

```

Τέλος μετά από κάθε εκτέλεση εμφανίζονται κατάλληλα μηνύματα κατάστασης της μνήμης και της διεργασίας.

```

void print_memory(){
    printf("\n---Memory Status---\n");
    for(int i = 0; i < num_blocks; ++i)
        printf("Block %d: Start = %d, Size = %d, isFree = %s, pID = %d\n",
            i, memory[i].start, memory[i].size, memory[i].isfree ? "yes"
: "no", memory[i].pid);
}

void print_process_status(Process processes[], int n){
    printf("\n---Process Status---\n");
    for(int i = 0; i < n; i++){
        printf("pID = %d, Arrival = %d, Duration = %d, Remaining = %d, Memory = %d, InMemory = %s\n",
            processes[i].pid, processes[i].arrival_time,
processes[i].duration,
            processes[i].remaining_time, processes[i].req_memory,
processes[i].in_memory ? "yes" : "no");
    }
}

```

Παρακάτω ακολουθούν μερικά παραδείγματα υλοποίησης του προγράμματος:

1) Θα εισάγουμε 2 διεργασίες όπου η 1η πιάνει όλο τον χώρο της μνήμης (512KB) ώστε να δούμε πως θα αντιμετωπίσει το πρόγραμμα το γεγονός ότι δεν θα χωράει η 2η διεργασία.

```
Enter amount of processes: 2
Enter pID, Arrival Time, Duration (Burst Time), Required Memory(KB):
1 1 3 512
Enter pID, Arrival Time, Duration (Burst Time), Required Memory(KB):
2 1 4 10

Process 1

---Memory Status---
Block 0: Start = 0, Size = 512, isFree = no, pID = 1

---Process Status---
pID = 1, Arrival = 1, Duration = 3, Remaining = 3, Memory = 512, InMemory = yes
pID = 2, Arrival = 1, Duration = 4, Remaining = 4, Memory = 10, InMemory = no
```

Βλέπουμε πως εισάγεται στη μνήμη η 1η διεργασία και η 2η όχι, όπως περιμέναμε. Βλέπουμε απο το Memory Status ότι υπάρχει 1 block που αποτελεί όλη τη μνήμη και κρατάει το process id (pID) της 1ης διεργασίας. Αντίστοιχα στο Process Status φαίνεται ότι δεν έχει εκτελεστεί ακόμη η 1η και βρίσκεται στη μνήμη, ενώ η 2η όχι.

```
---Process Status---
pID = 1, Arrival = 1, Duration = 3, Remaining = 0, Memory = 512, InMemory = no
pID = 2, Arrival = 1, Duration = 4, Remaining = 4, Memory = 10, InMemory = no

Process 2

---Memory Status---
Block 0: Start = 0, Size = 10, isFree = no, pID = 2
Block 1: Start = 10, Size = 502, isFree = yes, pID = -1

---Process Status---
pID = 1, Arrival = 1, Duration = 3, Remaining = 0, Memory = 512, InMemory = no
pID = 2, Arrival = 1, Duration = 4, Remaining = 4, Memory = 10, InMemory = yes

Process 2

---Memory Status---
Block 0: Start = 0, Size = 10, isFree = no, pID = 2
Block 1: Start = 10, Size = 502, isFree = yes, pID = -1

---Process Status---
pID = 1, Arrival = 1, Duration = 3, Remaining = 0, Memory = 512, InMemory = no
pID = 2, Arrival = 1, Duration = 4, Remaining = 1, Memory = 10, InMemory = yes

Process 2

---Memory Status---
Block 0: Start = 0, Size = 512, isFree = yes, pID = -1

---Process Status---
pID = 1, Arrival = 1, Duration = 3, Remaining = 0, Memory = 512, InMemory = no
pID = 2, Arrival = 1, Duration = 4, Remaining = 0, Memory = 10, InMemory = no
```

Απο πάνω βλέπουμε ότι εκτελέστηκε η 1η διεργασία απευθείας καθώς ο απαιτούμενος χρόνος εκτέλεσης ήταν ίσος με το κβάντο χρόνου για το Round Robin. Αμέσως μετά απελευθερώνεται η μνήμη, κάνει deallocate τον χώρο της 1ης διεργασίας και allocate για να εισαχθεί η 2η, όπως φαίνεται από το memory status δημιουργείται 1 block με την διεργασία 2 και το 2ο block μένει άδειο. Στη συνέχεια, λειτουργεί κανονικά ο αλγόριθμος Round Robin με 2 iterations για να εκτελέσει την 2η διεργασία (εφόσον ο απαιτούμενος χρόνος της ήταν 4ms > QT\_RoundRobin. Τέλος, απελευθερώνεται η μνήμη και επανέρχεται στο original block που είχε στο initialization.

Παρακάτω φαίνεται ένα παράδειγμα όπου οι διεργασίες έρχονται σε διαφορετικούς χρόνους:

```
mikemits@mike-vivobook:~/CEID/semester5/Λειτουργικά/PROJECT1/askisi3$ ./scheduler
Enter amount of processes: 2
Enter pID, Arrival Time, Duration (Burst Time), Required Memory(KB):
1 0 2 300
Enter pID, Arrival Time, Duration (Burst Time), Required Memory(KB):
2 1 2 100

[Time = 0] | Process 1 is scheduled.

---Memory Status---
Block 0: Start = 0, Size = 300, isFree = no, pID = 1
Block 1: Start = 300, Size = 212, isFree = yes, pID = -1

---Process Status---
pID = 1, Arrival = 0, Duration = 2, Remaining = 2, Memory = 300, InMemory = yes
pID = 2, Arrival = 1, Duration = 2, Remaining = 2, Memory = 100, InMemory = no

[Time = 2] | Process 1 is scheduled.

---Memory Status---
Block 0: Start = 0, Size = 512, isFree = yes, pID = -1

---Process Status---
pID = 1, Arrival = 0, Duration = 2, Remaining = 0, Memory = 300, InMemory = no
pID = 2, Arrival = 1, Duration = 2, Remaining = 2, Memory = 100, InMemory = no

[Time = 2] | Process 2 is scheduled.

---Memory Status---
Block 0: Start = 0, Size = 100, isFree = no, pID = 2
Block 1: Start = 100, Size = 412, isFree = yes, pID = -1

---Process Status---
pID = 1, Arrival = 0, Duration = 2, Remaining = 0, Memory = 300, InMemory = no
pID = 2, Arrival = 1, Duration = 2, Remaining = 2, Memory = 100, InMemory = yes

[Time = 4] | Process 2 is scheduled.

---Memory Status---
Block 0: Start = 0, Size = 512, isFree = yes, pID = -1

---Process Status---
pID = 1, Arrival = 0, Duration = 2, Remaining = 0, Memory = 300, InMemory = no
pID = 2, Arrival = 1, Duration = 2, Remaining = 0, Memory = 100, InMemory = no
```

## Ασκηση 4

Διεργασία	Χρόνος Άφιξης (XA)	Διάρκεια Εκτέλεσης (XE)	PID
A	0	6	3
B	2	4	1
Γ	3	1	2
Δ	4	3	5
E	5	5	4
Z	6	7	1

(α)

Διαγράμματα Gantt για:

- FCFS (First Come First Served).

Ο FCFS εκτελεί τις διεργασίες με βάση τον μικρότερο χρόνο άφιξης, δηλαδή ποιά ήρθε πρώτη όπως λέει και το όνομα του. Αρχικά η ουρά έχει τις A,B,Γ,Δ,E,Z και βγαίνει η A με XA = 0 και δεσμεύει την CPU για 6 χ.μ. . Μετά ακολουθεί η B με XA = 2, δηλ. έχει καθυστερήσει 4 χ.μ. και ξεκινά την στιγμή 6. Πλέον η ουρά είναι Γ,Δ,E,Z και την στιγμή 10 που τελειώνει η B μπαίνει η Γ και δεσμεύει τη CPU για μια χρονική μονάδα όσο ο XE της. Την στιγμή 11 μπαίνει η Δ και κρατά μέχρι την  $11+3=14$  χρονική στιγμή στην οποία μπαίνει η επόμενη στην ΚΜΕ, η E, με XE = 5 άρα θα διαρκέσει μέχρι τη στιγμή  $14+5 = 19$ . Τέλος εκτελείται η διαδικασία Z με XE = 7 άρα τελειώνει την εκτέλεση της την στιγμή 26.

A	B	Γ	Δ	E	Z	
0	6	10	11	14	19	26

- SJF (Shortest Job First).

Ο αλγόριθμος SJF πάλι σε αντιστοιχία με το όνομα του δρομολογεί τις διεργασίες με βάση την μικρότερη διάρκεια εκτέλεσης.

Με παρόμοια λογική υπολογίζουμε ότι οι διεργασίες θα έχουν τα εξής:



Διεργασία	Χρόνος Άφιξης (ΧΑ)	Διάρκεια Εκτέλεσης (ΧΕ)	Χρόνος εξόδου
A	0	6	$6 + 0 = 6$
B	2	4	$4 + 10 = 14$
Γ	3	1	$1 + 6 = 7$
Δ	4	3	$3 + 7 = 10$
E	5	5	$5 + 14 = 19$
Z	6	7	$19 + 7 = 26$

Θα περίμενε κανείς, λόγω της δρομολόγησης με βάση τον μικρότερο ΧΕ, να είναι η Γ η πρώτη εκτέλεση. Όμως η Γ έρχεται την στιγμή 3 και την στιγμή 0 η μόνη διεργασία στην ουρά είναι η Α η οποία ξεκινά. Μετά το πέρας της (στιγμή 6) έχουν πλέον έρθει όλες οι διεργασίες στην ουρά και περιμένουν, άρα τότε ξεκινάει ουσιαστικά ο αλγόριθμος στις διεργασίες μας. Αν και η Γ ερχόταν την στιγμή 0 τότε θα έμπαινε η Γ πρώτη.

Διάγραμμα Gantt:

A	Γ	Δ	B	E	Z	
0	6	7	10	14	19	26

- SRTF (Shortest Remaining Time First).

Ο αλγόριθμος SRTF μοιάζει με τον SJF μόνο που ελέγχει κάθε χ.μ. Για τυχόν αλλαγή στον εναπομείναντα ΧΕ, δηλαδή είναι ο SJF με preemption. Αρχικά την χρονική στιγμή 0 η μόνη διεργασία στην ουρά είναι η Α, όπως και την στιγμή 1.

Άρα:

Χρονική στιγμή 2:

Διεργασία	Remaining ΧΕ
A	4

Κάθε φορά που εκτελείται μια διεργασία για 1 χρονική μονάδα, μειώνεται κατά 1 ο εναπομείναντας ΧΕ της. Την στιγμή 2 έρχεται και η Β με ίδιο ΧΕ με την Α, όμως με βάση την εκφώνηση, για δύο διεργασίες που φτάνουν την ίδια στιγμή, εκείνη που άφησε την ΚΜΕ μπαίνει

πρώτη στην ουρά. Άρα συνεχίζει η A μέχρι την στιγμή 3, στην οποία έχει  $XE = 3$  ενώ η Γ που μόλις ήρθε έχει  $XE = 1$ . Άρα εκτελείται η Γ και η A μπαίνει μπροστά στην ουρά γιατί  $XE(A) < XE(B)$ .

Από την στιγμή 4, που τελειώνει η Γ, μέχρι την στιγμή 7 εκτελείται η A, γιατί παρόλο που έχει ίδιο  $XE$  με την Δ ( $= 3$ ), η οποία έφτασε την στιγμή 4, η A είναι πιο μπροστά στην ουρά. Την στιγμή 7 (τέλος της A), την μικρότερη διάρκεια εκτέλεσης έχει η Δ ( $= 3$ ) και εκτελείται μέχρι την στιγμή 10. Τότε εκτελείται η B, η οποία έχει μείνει στην ουρά με  $XE = 4$ , μέχρι την στιγμή 14. Τέλος, όπως και πριν εκτελούνται η E από την 14—19 και η Z από την 19—26.

Διάγραμμα Gantt:

A	A	A	Г	A	A	A	Δ	Δ	Δ	B	E	Z	
0	1	2	3	4	5	6	7	8	9	10	14	19	26

- RR (Round Robin) με κβάντο χρόνου 2 χρονικές μονάδες

Ο αλγόριθμος RR δίνει την ΚΜΕ σε κάθε διεργασία ανά περίοδο 2 χ.μ. για να υπάρχει η ψευδαίσθηση ότι την έχουν όλες πάντα. Επειδή δεν αναφέρεται στην εκφώνηση και τα αντίστοιχα παραδείγματα των διαφανειών έχουν κβάντο 1 (ή 10 με χρόνους που δεν προκύπτει διαφορά), άρα δεν πέφτουμε στην περίπτωση που θα δούμε παρακάτω, κάναμε την **παραδοχή ότι όταν τελειώνει μια διεργασία εντελώς, ανεξαρτήτως του κβάντου, έρχεται η επόμενη στην ΚΜΕ**. Θα μπορούσαμε να πάρουμε ότι αν τελειώσει καθολικά η εργασία τότε μπαίνει σε κατάσταση idle η ΚΜΕ όμως τότε θα αργούσε παραπάνω, κάτι που προφανώς είναι χειρότερο χρονικά από την παραδοχή μας.

Χρονική Στιγμή	Άφιξη	ΚΜΕ	Ουρά	Τερματισμός
0	A	A	-	-
2	B	A	B	-
4	Γ(3),Δ(4)	B	Γ,A,Δ	-
6	E(5),Z(6)	Γ	A,Δ,E,B,Z	-
7	-	A	Δ,E,Z,B,A	Γ
9	-	Δ	E,B,Z	A
11	-	E	B,Z,Δ	-
13	-	B	Z,Δ,E	-

15	-	Z	Δ,Ε	Β
17	-	Δ	Ε,Ζ	-
18	-	Ε	Ζ	Δ
20	-	Ζ	Ε	-
22	-	Ε	Ζ	-
23	-	Ζ	-	Ε
25	-	Ζ	-	-
26	-	-	-	Ζ

Διάγραμμα Gantt RR:

A	A	B	Γ	A	Δ	E	B	Z	Δ	E	Z	E	Z	Z	
0	2	4	6	7	9	11	13	15	17	18	20	22	23	25	26

(β)

Αρχικά για όλους τους αλγορίθμους ισχύει ότι:

- Μ.Χ. Αναμονής =  $[\text{Συνολικός (Χρόνος εξόδου - Χ.Α. - Χ.Ε.)} / 6]$  (πόση ώρα έκανε να εκτελεστεί πρώτη φορά από τον προκαθορισμένο Χ.Α.)
- Μ.Χ. Απόκρισης =  $[\text{Συνολικός (Χρόνος πρώτης εκκίνησης - Χ.Α.)} / 6]$
- Μ.Χ. Ολοκλήρωσης =  $[\text{Συνολικός (Χρόνος εξόδου - Χ.Α.)} / 6]$
- # Θεματικών αλλαγών = πόσες φορές άλλαξε διεργασία η ΚΜΕ

Και με εφαρμογή αυτών στα διαγράμματα Gantt έχουμε ότι:

- FCFS (First Come First Served)

Μ.Χ. Αναμονής:

$$(0+4+7+7+9+13)/6 = 6.7$$

Μ.Χ. Απόκρισης:

$$(0+4+7+7+9+13)/6 = 6.7$$

Μ.Χ. Ολοκλήρωσης:

$$(6+8+8+10+14+20)/6=11$$

# Θεματικών αλλαγών:

6 : A, A->B, B->Γ, Γ->Δ, Δ->E, E->Z

- SJF (Shortest Job First)

M.X. Αναμονής:

$$(0+8+3+3+9+13)/6 = 6$$

M.X. Απόκρισης:

$$(0+8+3+3+9+13)/6 = 6$$

M.X. Ολοκλήρωσης:

$$(6+12+4+6+14+20)/6=10.33$$

# Θεματικών αλλαγών:

6 : A, A->Γ, Γ->Δ, Δ->B, B->E, E->Z

- SRTF (Shortest Remaining Time First)

M.X. Αναμονής:

$$(1+8+0+3+9+13)/6 = 5.7$$

M.X. Απόκρισης:

$$(0+8+0+3+9+13)/6 = 5.5$$

M.X. Ολοκλήρωσης:

$$(7+12+1+6+14+20)/6=10$$

# Θεματικών αλλαγών:

7 : A, A->Γ, Γ->A, A->Δ, Δ->B, B->E, E->Z

- RR (Round Robin) με κβάντο χρόνου 2

M.X. Αναμονής:

$$(3+9+3+11+13+13)/6=8.7$$

M.X. Απόκρισης:

$$(0+2+3+5+6+9)/6=4.17$$

M.X. Ολοκλήρωσης:

$$(9+13+4+14+18+20)/6=13$$

# Θεματικών αλλαγών:

13 : A, A->B, B->Γ, Γ->A, A->Δ, Δ->E, E->B, B->Z, Z->Δ, Δ->E, E->Z, Z->E, E->Z

(Υ)

Ο αλγόριθμος LRTFP είναι η preemptive εκδοχή του LRTF. Πρώτα εξετάζουμε την συνθήκη μεγαλύτερου remaining χρόνου εκτέλεσης αλλά σε περιπτώσεις ισοδυναμίας αυτού προτεραιότητα έχει η διαδικασία με μικρότερο PID.

Ο αλγόριθμος ξεκινά τις πρώτες δύο χ.μ. Με τη διεργασία A η οποία είναι η μόνη στην ουρά. Στη στιγμή **2** η A έχει εκτελέσει 2 χ.μ. άρα τώρα έχει  $XE = 4$ . Την ίδια χρονική στιγμή φτάνει και η διεργασία B με επίσης  $XE = 4$ . Εδώ εφαρμόζεται η συνθήκη προτεραιότητας μικρότερου PID διότι ο XE τους είναι ο ίδιος, και εκτελείται πρώτη η B μέχρι την στιγμή **3**. Στην 3 φτάνει η διεργασία Γ με  $XE = 1$ , ο XE της B είναι πλέον 3 και της A είναι 4. Άρα εκτελείται η A μέχρι την στιγμή **4**. Τότε έρχεται η Δ στην ουρά με  $XE = 3 = XE(A) = XE(B)$  και λόγω αυτού εφαρμόζεται πάλι η συνθήκη προτεραιότητας του PID, άρα εκτελείται πάλι η B, και πλέον έχει  $XE = 2$ . Την χρονική στιγμή **5**, την άφιξη της κάνει η διεργασία E με  $XE = 5 >$  όλων των άλλων στην ουρά άρα και εκτελείται μέχρι την στιγμή **6** στην οποία έρχεται πρώτη φορά η Z(πλέον η E έχει  $XE = 4$ ). Η Z έχει  $XE = 7$  μεγαλύτερο από όλα άρα εκτελείται μέχρι την στιγμή **9** στην οποία έχει  $XE = 4 = XE(E)$  όμως η Z έχει μικρότερο PID άρα ξαναεκτελείται μέχρι την στιγμή **10**. Τότε εκτελείται η E με τον μεγαλύτερο  $XE = 4$  για μια χρονική μονάδα και την στιγμή **11** έχουμε ότι  $XE(Z)=XE(A)$ , άρα θα ξανα εκτελεστεί αυτή με το μικρότερο PID, η Z, μέχρι τη στιγμή **12**.

Μέχρι τη στιγμή 12 έχουμε ότι:

Διεργασία	Διάρκεια Εκτέλεσης (XE)	PID
A	3	3
B	2	1
Γ	1	2
Δ	3	5
E	3	4
Z	2	1

Με βάση τα PID και τους ΧΕ βλέπουμε ότι τις επόμενες 3 χρονικές στιγμές εκτελούνται με την εξής σειρά Α,Ε,Δ και την χρονική στιγμή **15** έχουν και οι 3 ΧΕ = 2.

Άρα:

Διεργασία	Διάρκεια Εκτέλεσης (ΧΕ)	PID
A	2	3
B	2	1
Γ	1	2
Δ	2	5
E	2	4
Z	2	1

Πλέον με βάση τα PID και τους ΧΕ πάλι ανα μία εκτελούνται με την σειρά Β,Ζ,Α,Ε,Δ. Διαλέξαμε την Β πριν από την Ζ ενώ έχουν το ίδιο PID γιατί ήταν πρώτη στο σύστημα. Άρα φτάνουμε στην στιγμή **20** και έχουμε ότι :

Διεργασία	Διάρκεια Εκτέλεσης (ΧΕ)	PID
A	1	3
B	1	1
Γ	1	2
Δ	1	5
E	1	4
Z	1	1

Βλέπουμε ότι όλες έχουν ΧΕ = 1 άρα κοιτάμε μόνο τα PIDs και καταλήγουμε στο ότι εκτελούνται ανα μία χ.μ. με την σειρά Β,Ζ,Γ,Α,Ε,Δ μέχρι την στιγμή **26**.

Το διάγραμμα Gantt είναι το εξής:

A	A	B	A	B	E	Z	Z	Z	Z	E	Z	A	E	Δ	B	Z	A	E	Δ	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

B	Z	Г	A	E	Δ	
20	21	22	23	24	25	26

Για τα ακόλουθα ερωτήματα ισχύουν τα ίδια με τα παραπάνω:

- M.X. Αναμονής:  
 $(18+15+19+19+15+9)/6 = 15.83$
- M.X. Απόκρισης:  
 $(0+0+19+10+0+0)/6=4.83$  , καθώς όλες οι εργασίες εκτελούνται στην ώρα τους εκτός από τις Γ και Δ
- M.X. Ολοκλήρωσης:  
 $(24+19+20+22+20+16)/6=20.17$
- # Θεματικών αλλαγών:  
22 :A, A->B , B->A , A->B , B->E, E->Z, Z->E, E->Z, Z->A, A->E, E->Δ, Δ->B, B->Z,  
Z->A, A->E, E->Δ, Δ->B, B->Z, Z->Γ, Γ->A, A->E, E->Δ