ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ 2017-18
1η Ατομική Άσκηση: Προσομοίωση Ουράς Τράπεζας

Να επεκτείνετε το πρόγραμμα προσομοίωσης της ουράς πελατών μιας Τράπεζας που συνοδεύει την άσκηση με νέες δυνατότητες και λειτουργικότητα ως ακολούθως

1. Να προσθέσετε στην υλοποίηση του ΑΤΔ Ουρά την πράξη
   int OuraGetSize(TOuras oura);
   που επιστρέφει τον αριθμό στοιχείων σε μια ουρά. Ελέγξτε την υλοποίηση με πρόγραμμα δοκιμής. Να επεκτείνετε το κυρίως πρόγραμμα, ώστε να εκτυπώνει τον αριθμό πελατών στην ουρά που δεν εξυπηρετήθηκαν μετά το τέλος της προσομοίωσης.
2. Να επεκτείνετε τον TStoixeiouOuras, ώστε το struct να περιλαμβάνει όχι μόνο το λεπτό εισόδου (όπως στην υλοποίηση που σας δίδεται), αλλά επίσης την διάρκεια εξυπηρέτησης ενός πελάτη (int XronosEksipiretisis που για την ώρα είναι σταθερός). Επίσης να προσθέσετε δυο συναρτήσεις
   void PelatisSetXronoEksipiretisis(TPelatis *stoixeioPtr, int duration);
   int PelatisGetXronoEksipiretisis(TPelatis *stoixeioPtr);
   Η πρώτη ενημερώνει τον χρόνο Εξυπηρέτησης και η δεύτερη επιστρέφει τον χρόνο εξυπηρέτησης. Ελέγξτε την υλοποίησή σας με πρόγραμμα δοκιμής και ενσωματώστε τις δυο συναρτήσεις στο κυρίως πρόγραμμα.
3. Κατόπιν να επεκτείνετε το κυρίως πρόγραμμα, ώστε αντί ο χρόνος εξυπηρέτησης ενός πελάτη να είναι σταθερός αυτός να είναι μεταξύ 0 (αποχώρηση πελάτη) και ενός μέγιστου, που να ζητείται ως είσοδος στο πρόγραμμα αντί του σταθερού χρόνου. Σημειώνουμε ότι η συνάρτηση rand( ) επιστρέφει τυχαίο ακέραιο μεταξύ 0 και MAX_RAND και επομένως rand( ) % M επιστρέφει τυχαίο μεταξύ 0 και M-1.
4. Να επεκτείνετε το κυρίως πρόγραμμα, ώστε να δέχεται τον χρόνο που κλείνει η τράπεζα (μετά τον οποίο δεν δέχεται άλλους πελάτες). Όμως οι υπάρχοντες πελάτες στην ουρά να εξυπηρετούνται όλοι. Να εκτυπώσετε τον πραγματικό τελικό χρόνο λειτουργίας και πόσα επιπλέον λεπτά ήταν απαραίτητα.
5. Να σχεδιάσετε και να αναπτύξετε μια νέα ενότητα για το ταμείο (tameio.h και tameio.c), που να ορίζει (struct TTameio) και να χειρίζεται α) τον χρόνο που ήταν απασχολημένο (TimeBusy), β) τον χρόνο που ήταν αδρανές (TimeInactive), γ) τον αριθμό πελατών που εξυπηρέτησε (ArithmoPelaton) και δ) τον εναπομείναντα χρόνο (enapomenonXronos) για να ολοκληρώσει την εξυπηρέτηση ενός πελάτη. Θα χρειαστείτε τις συναρτήσεις

```
void TameioDimiourgia(TTameio *tameio);                    //αρχικοποιεί τα μέλη του struct ταμείο
void TameioNewCustomer(TTameio *tameio, int Duration);
        // αυξάνει τον μετρητή πελατών και αρχικοποιεί εναπομείναντα χρόνο εξυπηρέτησης
void TameioInService(TTameio *tameio,);          // μειώνει χρόνο εξυπηρέτησης κατά 1
void TameioNoWork(TTameio *tameio);              // αυξάνει χρόνο αδράνειας κατά 1
void TameioBusy(TTameio *tameio);                // αυξάνει χρόνο απασχόλησης κατά 1
int TameioFree(TTameio tameio);                  // ελέγχει αν είναι διαθέσιμο

int TameioGetArithmosPelatwn(TTameio tameio);  // επιστρέφει τον αριθμό πελατών
int TameioGetInactiveXronos(TTameio tameio);   // επιστρέφει τον χρόνο αδράνειας του
int TameioGetBusyXronos(TTameio tameio);       // επιστρέφει τον χρόνο απασχόλησης του
```

   Να αλλάξετε το κυρίως πρόγραμμα, ώστε να χρησιμοποιεί αποκλειστικά τον ανωτέρω τύπο του Ταμείου (μέσω των συναρτήσεων). Δεν χρειάζεται αλλαγή δομής, αλλά αντί για την χρήση μεταβλητών προγράμματος να κάνετε χρήση των πράξεων της ενότητας ταμείο.
6. Να επεκτείνετε το κυρίως πρόγραμμα προσομοίωσης, ώστε να υποστηρίζει πολλά (σε πίνακα) ταμεία (οι πελάτες όπως πριν μπαίνουν σε μια κοινή ουρά). Το πρόγραμμα να δέχεται τον αριθμό ταμείων που έχει η Τράπεζα. Στο τέλος της προσομοίωσης να εκτυπώνει για κάθε ταμείο τον αριθμό πελατών που εξυπηρέτησε, τον χρόνο που ήταν απασχολημένο και τον χρόνο που ήταν αδρανές.
7. Για επιπλέον Bonus. Σχεδιάστε και αναπτύξτε, μια δική σας πολιτική διαχείρισης ταμείων, π.χ. να ξεκινάει ένα μόνο ταμείο ανοικτό και να βάζετε επιπλέον ταμείο όταν μεγαλώνει πολύ η ουρά και να κλείνετε το ταμείο όταν μικραίνει. Το πρόγραμμα θα πρέπει να έχει ως επιλογή την χρήση ή μη της πολιτικής αυτής, ώστε να φαίνεται η βελτίωση. Το bonus θα εφαρμοστεί εφόσον τα 1-6 έχουν απαντηθεί πλήρως.

Οδηγίες Σχεδίασης και Ανάπτυξης Προγράμματος
Σας δίδεται αρχικό πρόγραμμα που θα επεκτείνετε.
Το πρόγραμμά σας πρέπει να είναι οργανωμένο σε ενότητες (modules) και σε πρόγραμμα-πελάτη. Το πρόγραμμα πελάτης να μην χρησιμοποιεί άμεσα την δομή των ενοτήτων στα .h παρά μόνο μέσω των συναρτήσεων που ορίζονται.
Συνιστάται να κρατάτε αντίγραφα των προγραμμάτων σε κάθε στάδιο ανάπτυξης, π.χ. μετά την ολοκλήρωση κάθε ερωτήματος, ώστε να έχετε μια προηγούμενη έκδοση του προγράμματος σας. Ένα αντίγραφο θα σας φανεί πολύ χρήσιμο αν θέλετε να επιστρέψετε σε προηγούμενη έκδοση.

Παραδοτέα
1. Πηγαίος κώδικας (όλα τα .c, .h αρχεία σας). Επίσης το Makefile για gcc-linux. Σε κάθε αρχείο να αναφέρετε το όνομά σας στο εισαγωγικό σχόλιο.
2. Τεκμηρίωση σε σύντομο κείμενο (pdf) με την εξής δομή
   - Τα στοιχεία σας: (Όνομα-Επώνυμο-ΑΜ)
   - Λειτουργικότητα: Να περιγράψετε τι κάνει το πρόγραμμά σας (μπορεί να κάνει περισσότερα ή και λιγότερα από τα ζητούμενα της άσκησης). Ειδικά αν έχετε απαντήσει το ερώτημα 7, να αναφέρετε την πολιτική που σχεδιάσατε.
   - Οδηγίες Χρήσης του προγράμματος σας: π.χ. Διάταξη δεδομένων εισόδου.
   - Περιβάλλον Υλοποίησης και Δοκιμών: πχ. Αναπτύχθηκε σε gcc σε linux

Οδηγίες Παράδοσης
Το αρχείο τεκμηρίωσης μαζί με τα αρχεία του προγράμματος και το makefile να τα βάλετε σε έναν φάκελο, τον οποίο θα συμπιέσετε (zip, rar) και θα ανεβάσετε στο eclass.

Τρόπος Αξιολόγησης
Οι ασκήσεις είναι **ατομικές** και θα ελεγχθούν για ομοιότητες χρησιμοποιώντας ειδικό σύστημα εντοπισμού ομοιοτήτων/αντιγραφών. Σε περίπτωση μεγάλης «ομοιότητας» όλες οι «παρόμοιες» ασκήσεις θα μηδενιστούν και οι εμπλεκόμενοι θα αποκλειστούν από την τελική εξέταση.

Θα αξιολογηθούν η λειτουργικότητα, η δομή και η τεκμηρίωση του προγράμματος. Αναλυτικά:

Λειτουργικότητα (70/100)

1.	Πράξη OuraGetSize και αλλαγή main	(05)
2.	Επέκταση τύπου στοιχείου Pelati	(05)
3.	Επέκταση main με τυχαίο χρόνο εξυπηρέτησης	(05)
4.	Επέκταση προσομοίωσης με εξάντληση ουράς	(10)
5.	Τύπος στοιχείου Ταμείο και αλλαγές στο main	(25)
6.	Πολλά ταμεία και αλλαγές στο main	(20)
7.	Πολιτική Διαχείρισης Ταμείων	(10) Bonus

Δομή (25/100)

Οργάνωση σε Ενότητες (.h, .c) και πρόγραμμα πελάτη	(10)
Απόκρυψη Υλοποίησης, σωστή χρήση στο πρόγραμμα-πελάτης	(10)
Δομημένο Πρόγραμμα-πελάτη (μορφοποίηση, σχόλια, κλπ)	(05)

Τεκμηρίωση και Παρουσίαση (5/100)

ΠΡΟΣΟΧΗ:
Για να αξιολογηθεί το πρόγραμμα σας (έστω για την δομή του) πρέπει τουλάχιστον να μεταγλωττίζεται. Αν δεν μεταγλωττίζεται δεν παίρνει βαθμό. Πριν παραδώσετε το πρόγραμμά σας δοκιμάστε το μια τελευταία φορά και βεβαιωθείτε ότι παραδίδετε τα σωστά αρχεία.