1η Ατομική Άσκηση ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ 2016-17
Προσομοίωση Πολλαπλών Ουρών σε Super Market

Να μετατρέψετε/επεκτείνετε το πρόγραμμα προσομοίωσης της ουράς που συνοδεύει την άσκηση, με νέες δυνατότητες και λειτουργικότητα ως ακολούθως

1. Να προσθέσετε στον ΑΤΔ Ουρά την πράξη int OuraGetSize(TOuras oura) που επιστρέφει τον αριθμό στοιχείων Size που είναι μέσα στην ουρά. Να επεκτείνετε το κυρίως πρόγραμμα, ώστε να εκτυπώνει τον αριθμό πελατών στην ουρά που δεν εξυπηρετήθηκαν μετά το τέλος της προσομοίωσης.
2. Να προσθέσετε επίσης στον ΑΤΔ Ουρά ουρά δυο μετρητές CountIn, CountOut που να μετράνε πόσοι πελάτες μπήκαν στην ουρά και πόσοι βγήκαν από την ουρά συνολικά. Επίσης δυο αντίστοιχες συναρτήσεις OuraGetCountIn, OuraGetCountOut, που να επιστρέφουν τις τιμές των δυο μετρητών. Οι πράξεις πρόσθεση και απομάκρυνση του ΑΤΔ Ουρά να ελέγχουν (με assert) αν ισχύει η σταθερή συνθήκη (CountIn-CountOut==Size). Το κυρίως πρόγραμμα να εκτυπώνει στο τέλος τις τιμές CountIn και CountOut.
3. Να επεκτείνετε τον TStoixeiouOuras, ώστε το struct να περιλαμβάνει όχι μόνο τον χρόνο εισόδου (όπως στην υλοποίηση που σας δίδεται), αλλά επίσης την διάρκεια εξυπηρέτησης του πελάτη σε λεπτά (int XronosEksipiretisis για την ώρα είναι σταθερός). Επίσης να προσθέσετε δυο συναρτήσεις
   ```
   void PelatisSetXronoEksipiretisis (TSOuras *stoixeioPtr, int duration);
   int   PelatisGetXronoEksipiretisis (TSOuras stoixeio);
   ```
   Η πρώτη συνάρτηση ενημερώνει τον χρόνο εξυπηρέτησης duration για κάθε πελάτη όταν μπαίνει στην ουρά και η δεύτερη τον επιστρέφει.
   Κατόπιν να επεκτείνετε το κυρίως πρόγραμμα, ώστε, αντί ο χρόνος εξυπηρέτησης ενός πελάτη να είναι σταθερός, αυτός να είναι τυχαίος μεταξύ ενός Ελάχιστου (π.χ. 1 λεπτό) και ενός Μέγιστου (πχ. 10 λεπτά) που να ζητούνται ως είσοδοι στο πρόγραμμα αντί του σταθερού χρόνου εξυπηρέτησης. Σημειώνουμε ότι η συνάρτηση rand( ) επιστρέφει τυχαίο ακέραιο μεταξύ 0 και MAX_RAND, επομένως rand( ) % N επιστρέφει τυχαίο μεταξύ 0 και N-1.
4. Να σχεδιάσετε και να αναπτύξετε τον ΑΤΔ Ταμία, που να περιλαμβάνει την δήλωση του τύπου του (typedef struct {…} TTamias) με μέλη του struct α) τον εναπομείναντα χρόνο (enapomenonXronos) για να ολοκληρώσει την εξυπηρέτηση ενός πελάτη β) τον αθροιστικό χρόνο που ο ταμίας είναι απασχολημένος (TimeBusy), γ) τον αθροιστικό χρόνο που ήταν αδρανής (TimeInactive) και δ) τον αριθμό πελατών που εξυπηρέτησε (ArithmoPelaton). Να αναπτύξετε αντίστοιχες συναρτήσεις διαχείρισης του TTamias
   ```
   void TamiasDimiourgia(TTamias *Tamias);            // αρχικοποιεί τα μέλη του struct
   void TamiasNewCustomer(TTamias *Tamias);
         // αυξάνει κατά 1 τον μετρητή πελατών που έχει εξυπηρετήσει
   void TamiasSetXrono(TTamias *Tamias, int xronosEksipiretisis);
      // αρχικοποιεί εναπομείναντα χρόνο ως τον χρόνο εξυπηρέτησης πελάτη
   void TamiasSetBusy(TTamias *Tamias);
   // αυξάνει τον  χρόνο απασχόλησης του και μειώνει εναπομείνανα χρόνο κατά 1 μονάδα χρόνου
   void TamiasNoWork(TTamias *Tamias);     // αυξάνει χρόνο αδράνειας κατά 1 μονάδα χρόνου

   int TamiasFree(TTamias Tamias);                          // ελέγχει αν είναι διαθέσιμος
   int TamiasGetArithmosPelatwn(TTamias Tamias);    // επιστρέφει αριθμό πελατών που εξυπηρέτησε
   int TamiasGetEnapomenonXronos(TTamias Tamias); // επιστρέφει εναπομείναντα χρόνο
   int TamiasGetInactiveXronos(TTamias Tamias);        // επιστρέφει χρόνο αδράνειας
   int TamiasGetBusyXronos(TTamias Tamias);            // επιστρέφει χρόνο απασχόλησης
   ```
   Να αλλάξτε το κυρίως πρόγραμμα, ώστε να χρησιμοποιεί αποκλειστικά τον ανωτέρω τύπο του Ταμία (μέσω των συναρτήσεων). Δεν χρειάζεται αλλαγή δομής στο if μέσα στο κεντρικό while, αλλά αντί για την χρήση μεταβλητών να κάνετε χρήση των πράξεων του ΑΤΔ Tamias.
5. Να επεκτείνετε το κυρίως πρόγραμμα προσομοίωσης ώστε να υποστηρίζει Κ ουρές, που η καθεμία να έχει τον δικό της ταμία. Οι πελάτες μπαίνουν στις ουρές κυκλικά ούτως ώστε να μοιράζονται. Στο τέλος της προσομοίωσης το κυρίως πρόγραμμα να εκτυπώνει για κάθε ελεγκτή τον αριθμό πελατών που εξυπηρέτησε, τον χρόνο που ήταν απασχολημένος, τον χρόνο που είναι αδρανής και τον αριθμό των

πελατών που έχουν μείνει σε κάθε ουρά. Για κάθε ουρά τα CountIn, CountOut και πόσα δεν εξυπηρετήθηκαν. Να Ελέγξετε την σχέση των τιμών αυτών με assert.

6. Για επιπλέον Bonus 10%. Να επεκτείνετε το κυρίως πρόγραμμα, ώστε να δέχεται τον χρόνο που κλείνει το super market μετά τον οποίο δεν δέχεται άλλους πελάτες. Όμως οι υπάρχοντες πελάτες στο κατάστημα να εξυπηρετούνται όλοι. Δηλαδή όλοι οι πελάτες που μπαίνουν στο κατάστημα να μπαίνουν σε ουρά και να εξυπηρετούνται. Να εκτυπώσετε τον πραγματικό τελικό χρόνο λειτουργίας των ταμείων και πόσα επιπλέον λεπτά ήταν απαραίτητα από το κλείσιμο του καταστήματος.

7. Οδηγίες Σχεδίασης και Ανάπτυξης Προγράμματος

Το πρόγραμμά σας 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. Σας δίδεται αρχικό πρόγραμμα που θα επεκτείνετε. Στο eclass μπορείτε να βρείτε πρόγραμμα δοκιμής ουράς. Συστήνω να ελέγξετε την υλοποίηση της ουράς με πρόγραμμα δοκιμής.
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 ή το project file του DevC++. Σε κάθε αρχείο να αναφέρετε το όνομά σας και ΑΜ στο εισαγωγικό σχόλιο.
2. Τεκμηρίωση (μέγιστο 1 σελίδα) 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 6, να αναφέρετε τον σχεδιασμό της λύσης.
   - Οδηγίες Χρήσης του προγράμματος σας: π.χ. Διάταξη δεδομένων εισόδου.
   - Περιβάλλον Υλοποίησης και Δοκιμών: πχ. Αναπτύχθηκε σε Dev C++ σε περιβάλλον Windows 10, δοκιμάστηκε επίσης σε gcc σε Linux.

Οδηγίες Παράδοσης

Τα παραδοτέα (αρχείο τεκμηρίωσης και τα αρχεία του προγράμματος) να τα βάλετε σε έναν φάκελο, τον οποίο θα συμπιέσετε (zip, rar) με όνομα τον αριθμό μητρώου σας και θα ανεβάσετε στο eclass. Προσοχή: Να το ανεβάσετε στην κατάλληλη κατηγορία υλοποίησης (Dev-C++ ή gcc).

Τρόπος Αξιολόγησης

Οι ασκήσεις είναι **ατομικές** και θα ελεγχθούν για ομοιότητες χρησιμοποιώντας ειδικό σύστημα εντοπισμού ομοιοτήτων/αντιγραφών. Σε περίπτωση μεγάλης «ομοιότητας» οι εμπλεκόμενοι αποκλείονται από τις επόμενες ασκήσεις, τα εργαστήρια και την τελική εξέταση. Θα αξιολογηθούν η λειτουργικότητα, η δομή και η τεκμηρίωση του προγράμματος. Αναλυτικά:

Λειτουργικότητα (70/100)+10 bonus

1.	Πράξη SizeOuras και αλλαγή main	(05)
2.	CountIn, CountOut + assert και αλλαγή στο main	(10)
3.	Επέκταση τύπου στοιχείου ουράς και μεταβλητού χρόνου εξυπηρέτησης	(15)
4.	Τύπος στοιχείου Tamias και αλλαγές στο main	(25)
5.	Πολλοί Ταμίες και Ουρές + assert	(15)
6.	Κλείσιμο Καταστήματος και εξυπηρέτηση	(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 (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.