ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ 2015-16
1η Ατομική Άσκηση: Προσομοίωση Πολλαπλών Ουρών Διοδίων

Να επεκτείνετε το πρόγραμμα προσομοίωσης της ουράς Διοδίων («πελάτης» στο πρόγραμμα δηλώνει αυτοκίνητο), 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. Να επεκτείνετε το κυρίως πρόγραμμα, ώστε να εκτυπώνει τον αριθμό πελατών/αυτοκινήτων στην ουρά που δεν εξυπηρετήθηκαν μετά το τέλος της προσομοίωσης.
2. Να προσθέσετε στην ουρά δυο μετρητές CountIn, CountOut που να μετράνε πόσα αυτοκίνητα μπήκαν και πόσα βγήκαν συνολικά. Επίσης δυο αντίστοιχες συναρτήσεις GetCountIn, GetCountOut, που να επιστρέφουν τις τιμές των μετρητών. Το κυρίως πρόγραμμα να εκτυπώνει τις τιμέε αυτές.
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 (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 για κάθε αυτοκίνητο και η δεύτερη επιστρέφει τον χρόνο εξυπηρέτησης. Κατόπιν να επεκτείνετε το κυρίως πρόγραμμα, ώστε, αντί ο χρόνος εξυπηρέτησης ενός πελάτη να είναι σταθερός, αυτός να είναι τυχαίος μεταξύ ενός Ελάχιστου και ενός Μέγιστου που να ζητούνται ως είσοδοι στο πρόγραμμα αντί του σταθερού χρόνου. Σημειώνουμε ότι η συνάρτηση rand( ) επιστρέφει τυχαίο ακέραιο μεταξύ 0 και MAX_RAND, rand( ) % X επιστρέφει τυχαίο μεταξύ 0 και X-1.
4. Να σχεδιάσετε και να αναπτύξετε έναν ΑΤΔ για τον ελεγκτή. Ο νέος ΑΤΔ να ονομάζεται Controller και πρέπει να περιλαμβάνει δηλώσεις για τον ελεγκτή, (typedef struct {…} TController) και να χειρίζεται α) τον χρόνο που ήταν απασχολημένος (TimeBusy), β) τον χρόνο που ήταν αδρανής (TimeInactive), γ) τον αριθμό πελατών/αυτοκινήτων που εξυπηρέτησε (ArithmoPelaton) και δ) τον εναπομείναντα χρόνο (enapomenonXronos) για να ολοκληρώσει την εξυπηρέτηση ενός πελάτη. Να αναπτύξετε αντίστοιχες συναρτήσεις διαχείρισης των ανωτέρω
   ```
   void ControllerDimiourgia(TController *Controller);          // αρχικοποιεί τα μέλη του struct
   void ControllerNewCustomer(TController *Controller);
         // αυξάνει τον μετρητή πελατών και προσθέτει 1 μονάδα χρόνου απασχόλησης του ελεγκτή
   void ControllerSetXrono(TController *Controller, int xronosEksipiretisis);
      // αρχικοποιεί εναπομείναντα χρόνο ως τον χρόνο εξυπηρέτησης για κάθε νέο πελάτη-αυτοκίνητο
   void ControllerNoWork(TController *Controller);   // αυξάνει χρόνο αδράνειας κατά 1 μονάδα
   void ControllerBusy(TController *Controller);
         // αυξάνει χρόνο απασχόλησης και μειώνει εναπομείνανα χρόνο κατά 1 μονάδα
   int ControllerFree(TController Controller);                    // ελέγχει αν είναι διαθέσιμος
   int ControllerGetArithmosPelatwn(TController Controller);      // επιστρέφει αριθμό πελατών
   int ControllerGetEnapomenonXronos(TController Controller); // επιστρέφει εναπομείναντα χρόνο
   int ControllerGetInactiveXronos(TController Controller);       // επιστρέφει χρόνο αδράνειας
   int ControllerGetBusyXronos(TController Controller);           // επιστρέφει χρόνο απασχόλησης
   ```

   Να αλλάξτε το κυρίως πρόγραμμα, ώστε να χρησιμοποιεί αποκλειστικά τον ανωτέρω τύπο του Ελεγκτή (μέσω των συναρτήσεων). Δεν χρειάζεται αλλαγή δομής στο if, αλλά αντί για την χρήση μεταβλητών προγράμματος να κάνετε χρήση των πράξεων του ΑΤΔ Controller.
5. Να επεκτείνετε το κυρίως πρόγραμμα προσομοίωσης ώστε να υποστηρίζει πολλές ουρές, που η καθεμία να έχει τον δικό της ελεγκτή. Τα αυτοκίνητα επιλέγουν να μπουν στην ουρά με τα λιγότερα αυτοκίνητα. Στο τέλος της προσομοίωσης να εκτυπώνει για κάθε ελεγκτή τον αριθμό πελατών που εξυπηρέτησε, τον χρόνο που ήταν απασχολημένος και τον χρόνο που είναι αδρανής και τα αυτοκίνητα που έχουν μείνει σε κάθε ουρά. Για κάθε ουρά τα CountIn, CountOut και πόσα δεν εξυπηρετήθηκαν.
6. Για επιπλέον Bonus 10%. Σχεδιάστε και αναπτύξτε μια δική σας πολιτική διαχείρισης ελεγκτών, π.χ. να ανοίγουν και να κλείνουν δυναμικά ελεγκτήρια ανάλογα με το μέγεθος των ουρών και την αδράνεια των ελεγκτών.

Οδηγίες Σχεδίασης και Ανάπτυξης Προγράμματος
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. 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 την πολιτική που σχεδιάσατε.
   - Οδηγίες Χρήσης του προγράμματος σας: π.χ. Διάταξη δεδομένων εισόδου.
   - Περιβάλλον Υλοποίησης και Δοκιμών: πχ. Αναπτύχθηκε σε Dev C++ σε περιβάλλον Windows XP, δοκιμάστηκε επίσης σε gcc σε Linux

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

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

Λειτουργικότητα (70/100)

1.	Πράξη SizeOuras και αλλαγή main	(05)
2.	CountIn, CountOut και αλλαγή στο main	(10)
3.	Επέκταση τύπου στοιχείου ουράς και τυχαίου χρόνου εξυπηρέτησης	(15)
4.	Τύπος στοιχείου Controller και αλλαγές στο main	(25)
5.	Πολλοί Controllers και Ουρές	(15)
6.	Πολιτική Διαχείρισης Ελεγκτών-Ουρών	(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.