ΕΡΓΑΣΤΗΡΙΟ 7

Στο προηγούμενο εργαστήριο υλοποιήσαμε ένα πρόγραμμα που λύνει το πρόβλημα του Josephus, χρησιμοποιώντας μια ουρά. Όπως τονίσαμε στην διάρκεια του εργαστηρίου, σε μια πρώτη φάση, έπρεπε να υλοποιήσουμε τις συναρτήσεις σχετικά με την ουρά (qEnqueue(), qDequeue(), isEmpty(), κλπ). Μετά, υλοποιήσαμε τον αλγόριθμο που λύνει το πρόβλημα χρησιμοποιώντας την δομή της ουράς. Όλα αυτά έγιναν στο αρχείο josephus.c. Εάν χρειαστεί να υλοποιήσουμε άλλο αλγόριθμο χρησιμοποιώντας την ίδια δομή, πρέπει να ξαναγράψουμε το ίδιο κώδικα για την δομή, πράγμα όχι και τόσο βολικό. Γι' αυτό τον λόγο, αλλά και για άλλους τους οποίους έχετε συζητήσει στις διαλέξεις, είναι προτιμότερο να διασπάσουμε τον κώδικα σε πολλά αρχεία.

Στις παρακάτω ασκήσεις πρέπει:

- Να διασπάσετε τον κώδικα του προγράμματος σε πολλαπλά αρχεία.
- Να χρησιμοποιήσετε ένα makefile για να μεταγλωττίσετε το πρόγραμμα αποτελούμενο τώρα από πολλά αρχεία.
- Προσθέστε ένα driver αντικειμένου στο πρόγραμμά σας.

1. Γράψτε ένα πρόγραμμα infixTOpostfix.c που μετατρέπει μια αριθμητική έκφραση από infix μορφή σε postfix μορφή. Π.χ.

   Έκφραση σε infix μορφή: ***(2 + ((3 + 4) * (5 * 6)))***
   Ανάλογη έκφραση σε postfix μορφή: ***2 3 4 + 5 6 * * +***

   Όταν δίνετε την έκφραση σε infix μορφή, χρησιμοποιήστε παρενθέσεις για να καθορίσετε την προτεραιότητα των πράξεων.

2. Γράψτε ένα πρόγραμμα το οποίο διαβάζει μια αριθμητική έκφραση ή μια γραμμή κειμένου από τον χρήστη, και ελέγχει εάν οι παρενθέσεις/αγκιλες είναι ισορροπημένες, π.χ. το πρόγραμμα πρέπει να επιστρέψει `true` για την είσοδο: `[()]{}{[()()]()}` και false για: `[(])`. Το πρόγραμμα να χρησιμοποίει μια στοίβα.