ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ

Τμήμα Εφαρμοσμένης Πληροφορικής

ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ

Εξάμηνο Β'

Φύλλο Ασκήσεων 3: ΟΥΡΕΣ (υλοποίηση με πίνακες)

Μάγια Σατρατζέμη, Γεωργία Κολωνιάρη, Αλέξανδρος Καρακασίδης

Παρατηρήσεις:

1. *Τα δεδομένα εισόδου διαβάζονται πάντα με ξεχωριστές εντολές scanf( ) το καθένα και με τη σειρά που δηλώνονται στις εκφωνήσεις.*
2. *Αντίστοιχα για τα δεδομένα εξόδου και όπου δεν υπάρχουν περαιτέρω διευκρινήσεις για τη μορφή τους, αυτά θα εμφανίζονται με ξεχωριστές εντολές printf(" ..\n") το καθένα και με τη σειρά που δηλώνονται στις εκφωνήσεις.*
   i) *Τα στοιχεία των κόμβων της ουράς θα εμφανίζονται σε μια γραμμή με ένα κενό χαρακτήρα μεταξύ τους. Σε περίπτωση που οι κόμβοι της ουράς περιέχουν περισσότερα από ένα στοιχεία, τότε τα στοιχεία κάθε κόμβου θα εμφανίζονται σε μια γραμμή με ένα κενό χαρακτήρα μεταξύ τους, ενώ κάθε κόμβος θα εμφανίζεται σε διαφορετική γραμμή.*
3. *Σε όσες από τις ασκήσεις θεωρείται δεδομένη η ύπαρξη ουράς θα πρέπει προηγουμένως να τη δημιουργήσετε.*
4. ***ΠΡΟΣΟΧΗ: Οι ασκήσεις θα πρέπει να λύνονται με χρήση του κώδικα που υλοποιεί τον ΑΤΔ ουρά. Ο κώδικας που σας δίνεται περιλαμβάνεται στο code.zip στην αντίστοιχη διάλεξη. Οι συναρτήσεις που υλοποιούν τις βασικές λειτουργίες του ΑΤΔ ουρά δεν τροποποιούνται. Τροποποιήσεις μπορούν να γίνουν ανάλογα με τη άσκηση και εφόσον χρειάζεται μόνο στο πλήθος των στοιχείων της ουράς και στον τύπο του στοιχείου της ουράς και στη συνάρτηση TraverseQ (είναι βοηθητική συνάρτηση, δεν υλοποιεί λειτουργία του ΑΤΔ ουρά).***

---

1. Γράψτε πρόγραμμα που θα περιλαμβάνει και θα καλεί σειριακά τις παρακάτω συναρτήσεις:

- Συνάρτηση *QSizeA* που θα υπολογίζει το πλήθος των στοιχείων μιας ουράς *Q* χρησιμοποιώντας τις πληροφορίες που περιλαμβάνονται στην εγγραφή και τον πίνακα που χρησιμοποιούνται για την αποθήκευση της ουράς.
- Συνάρτηση QSizeB που θα υπολογίζει το πλήθος των στοιχείων μιας ουράς *Q* χρησιμοποιώντας μόνο τις λειτουργίες επιπέδου εφαρμογής *CreateQ*, *EmptyQ*, *AddQ* και *RemoveQ*. Η συνάρτηση πρέπει να έχει την ακόλουθη επικεφαλίδα:

```
int QSizeB(QueueType *Q);
/* Δέχεται:       μία ουρά Q
   Λειτουργία:    υπολογίζει το πλήθος των στοιχείων της ουράς Q
   Επιστρέφει:    το πλήθος των στοιχείων της ουράς Q. */
```

Στο κυρίως πρόγραμμα θα καλούνται, όπως ήδη αναφέρθηκε, οι συναρτήσεις σειριακά και θα εμφανίζεται η τιμή που επιστρέφουν σε κάθε μία από τις παρακάτω περιπτώσεις:

(a) (*η ουρά Q να είναι γεμάτη*). Η ουρά που θα δημιουργήσετε θα περιλαμβάνει 20 αριθμούς. Για λόγους απλότητας μπορεί να χρησιμοποιηθεί ένας βρόχος for, σε κάθε επανάληψη του οποίου θα προστίθεται η τιμή της μεταβλητής ελέγχου στην ουρά. Μετά την εισαγωγή όλων των παραπάνω στοιχείων θα εμφανίζετε τα στοιχεία της ουράς (χρήση της βοηθητικής συνάρτησης TraverseQ) και στη συνέχεια θα καλέσετε τις QSizeA και QSizeB. Μετά την κλήση της QSizeA και QSizeB θα εμφανίζετε κάθε φορά τα στοιχεία της ουράς

(b) (*η ουρά Q περιλαμβάνει 10 στοιχεία*). Θα γεμίσετε εκ νέου την ουρά με 20 στοιχεία στη συνέχεια θα διαγράψετε τα 10 πρώτα κατά σειρά στοιχεία. Στη συνέχεια θα εμφανίζετε τα στοιχεία της ουράς (χρήση της βοηθητικής συνάρτησης TraverseQ) και στη συνέχεια θα καλέσετε τις QSizeA και QSizeB. Μετά την κλήση της QSizeA και QSizeB θα εμφανίζετε κάθε φορά τα στοιχεία της ουράς

(c) (*η ουρά Q είναι κενή*). Θα καλέσετε τις QSizeA και QSizeB. Μετά την κλήση της QSizeA και QSizeB θα εμφανίζετε κάθε φορά τα στοιχεία της ουράς

Η έξοδος του προγράμματος:

```
Question a
Printing Queue
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
(QSizeA) size=20
Printing Queue
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
(QSizeB) size=20
Printing Queue
```

```
Empty Queue

Question b
Printing Queue
10 11 12 13 14 15 16 17 18 19
(QSizeA) size=10
Printing Queue
10 11 12 13 14 15 16 17 18 19
(QSizeB) size= 10
Printing Queue
Empty Queue

Question c
(QSizeA) size=0
Printing Queue
Empty Queue
(QSizeB) size=0
Printing Queue
Empty Queue
```

2. Γράψτε μια συνάρτηση που εμφανίζει τα περιεχόμενα μιας ουράς *Q* τύπου *QueueType* την οποία δέχεται μέσω παραμέτρου. Η εμφάνιση θα γίνεται ξεκινώντας από την εμπρός άκρη της ουράς, ενώ η ουρά θα παραμένει αναλλοίωτη. Υλοποιήστε δύο εκδόσεις της συνάρτησης:

(a) υλοποιήστε τη συνάρτηση *DisplayQA(QueueType *Q)* σε επίπεδο εφαρμογής: μόνο οι λειτουργίες *AddQ* και *RemoveQ* μπορούν να χρησιμοποιηθούν.

(b) υλοποιήστε τη συνάρτηση *DisplayQB(QueueType Q)* σε επίπεδο υλοποίησης: μπορείτε να προσπελάσετε άμεσα τη δομή που χρησιμοποιείται για την αποθήκευση της ουράς χωρίς να κάνετε χρήση των διαδικασιών *AddQ* και *RemoveQ*. (*πρόκειται για τη TraverseQ που δίνεται στο Project_QueueADT*)

Οι δύο διαδικασίες θα καλούνται από το κυρίως πρόγραμμα σειριακά. Για να ελέγξετε την ορθότητα των δύο διαδικασιών δημιουργήστε προηγουμένως στο κυρίως πρόγραμμα μια ουρά που περιλαμβάνει όλους τους περιττούς αριθμούς στο διάστημα [1..100].

3. Γράψτε συναρτήσεις που επιστρέφουν αντίστοιχα το στοιχείο της εμπρός, και της πίσω άκρης μιας ουράς *Q* τύπου *QueueType*. Αν η ουρά είναι κενή θα επιστρέφεται η τιμή -1. Υλοποιήστε δύο εκδόσεις της συνάρτησης για κάθε μία από τις προαναφερθείσες λειτουργίες:

(a) υλοποιήστε τη συνάρτηση *GetFrontElementA* για την επιστροφή του στοιχείου της εμπρός άκρης της ουράς σε επίπεδο εφαρμογής: μόνο οι λειτουργίες *EmptyQ*, *AddQ* και *RemoveQ* μπορούν να χρησιμοποιηθούν.

(b) υλοποιήστε τη συνάρτηση *GetFrontElementB* για την επιστροφή του στοιχείου της εμπρός άκρης της ουράς σε επίπεδο υλοποίησης: μπορείτε να προσπελάσετε άμεσα τη δομή που χρησιμοποιείται για την αποθήκευση της ουράς χωρίς να κάνετε χρήση των διαδικασιών *AddQ* και *RemoveQ*.

(c) υλοποιήστε τη συνάρτηση *GetRearElementB* για την επιστροφή του στοιχείου της πίσω άκρης της ουράς σε επίπεδο υλοποίησης: μπορείτε να προσπελάσετε άμεσα τη δομή που χρησιμοποιείται για την αποθήκευση της ουράς χωρίς να κάνετε χρήση των διαδικασιών *AddQ* και *RemoveQ*.

(d) υλοποιήστε τη συνάρτηση *GetRearElementA* για την επιστροφή του στοιχείου της πίσω άκρης της ουράς σε επίπεδο εφαρμογής: μόνο οι λειτουργίες *EmptyQ*, *AddQ* και *RemoveQ* μπορούν να χρησιμοποιηθούν.

Οι παράμετροι των παραπάνω συναρτήσεων θα είναι 2: η ουρά και το στοιχείο (εμπρός ή πίσω άκρη της ουράς) αντίστοιχα. Οι συναρτήσεις θα είναι void.

Οι συναρτήσεις θα καλούνται από το κυρίως πρόγραμμα σειριακά και θα εμφανίζεται η τιμή που επιστρέφουν, εφόσον αυτή είναι διαφορετική του -1. Για να ελέγξετε την ορθότητα των δύο διαδικασιών δημιουργήστε προηγουμένως στο κυρίως πρόγραμμα μια ουρά που περιλαμβάνει τα πολλαπλάσια του 3 στο διάστημα [1..50]. Μετά την εισαγωγή των στοιχείων στην ουρά και μετά από την κλήση των συναρτήσεων a) – d) καλέστε την TraverseQ. Στη συνέχεια δίνεται ένα στιγμιότυπο εκτέλεσης:

```
after AddQ: 3 6 9 12 15 18 21 24 27 30 33 36 39 42 45 48
a) -> 3
after GetFrontElementA: 6 9 12 15 18 21 24 27 30 33 36 39 42 45 48
b) -> 6
after GetFrontElementB: 6 9 12 15 18 21 24 27 30 33 36 39 42 45 48
c) -> 48
after GetRearElementB: 6 9 12 15 18 21 24 27 30 33 36 39 42 45 48
d) -> 48
after GetRearElementA:
Πιέστε ένα πλήκτρο για συνέχεια. . .
```

4. Γράψτε μια συνάρτηση *GetNthElement* που δέχεται μια ουρά τύπου *QueueType* και έναν ακέραιο αριθμό *n* και επιστρέφει το *n*-οστό στοιχείο της ουράς. Υλοποιήστε δύο εκδόσεις της συνάρτησης:

(a) υλοποιήστε τη συνάρτηση *GetNthElementA* σε επίπεδο εφαρμογής: μόνο οι λειτουργίες *EmptyQ*, *AddQ* και *RemoveQ* μπορούν να χρησιμοποιηθούν.

(b) υλοποιήστε τη συνάρτηση *GetNthElementB* σε επίπεδο υλοποίησης: μπορείτε να προσπελάσετε άμεσα τη δομή που χρησιμοποιείται για την αποθήκευση της ουράς χωρίς να κάνετε χρήση των διαδικασιών *AddQ* και *RemoveQ*.

Στο κυρίως πρόγραμμα θα διαβάζεται το *n*, θα καλούνται οι δύο διαδικασίες και θα εμφανίζεται η τιμή που επιστρέφουν. Το *n* πρέπει να έχει τιμή μικρότερη ή ίση του πλήθους των στοιχείων της ουράς και θα ζητείται κατ' επανάληψη από τον χρήστη μέχρι να δοθεί σωστή τιμή (*n<= πλήθος στοιχείων ουράς*). Το πλήθος των στοιχείων της ουράς μπορεί να υπολογιστεί χρησιμοποιώντας τις πληροφορίες της δομής όπου αποθηκεύεται η ουρά. Για να ελέγξετε την ορθότητα των δύο διαδικασιών δημιουργήστε στο κυρίως πρόγραμμα μια ουρά που περιλαμβάνει τα πολλαπλάσια του 5 στο διάστημα [1..100].

5. Να γράψετε πρόγραμμα που θα περιλαμβάνει μια συνάρτηση *ReverseQ*(*QueueType *Queue)* που αντιστρέφει τα στοιχεία μιας ουράς χρησιμοποιώντας τις βασικές λειτουργίες που ορίζονται στους ΑΤΔ στοίβα και ουρά με πίνακες (*θα χρειασθεί και στοίβα για να αντιστρέψουμε τα στοιχεία της ουράς*). Για να ελέγξετε την ορθότητα του προγράμματος σας δημιουργήστε μια ουρά που περιλαμβάνει όλα τα πολλαπλάσια του 2 στο διάστημα [1..30]. Η ουρά θα εμφανιστεί πριν και μετά την αντιστροφή των στοιχείων της (κάντε χρήση της βοηθητικής συνάρτησης TraverseQ).

6. Μια εναλλακτική υλοποίηση μιας ουράς που χρησιμοποιεί ένα κυκλικό πίνακα και δεν απαιτεί να διατηρούμε μία κενή θέση μεταξύ της εμπρός και της πίσω άκρη της για να ξεχωρίζει μια κενή ουρά από μια γεμάτη χρειάζεται απλά την προσθήκη ενός ακέραιου πεδίου *Count* στην εγγραφή τύπου *QueueType*, στο οποίο αποθηκεύεται ο τρέχων αριθμός στοιχείων της ουράς. Να κάνετε τις απαραίτητες αλλαγές στη δήλωση του τύπου της εγγραφής και στις βασικές λειτουργίες του ΑΤΔ ουρά με πίνακες, έτσι ώστε να χρησιμοποιείται αυτό το επιπλέον πεδίο και να μην διατηρείται κενή θέση στον πίνακα όπου αποθηκεύονται τα στοιχεία της ουράς. Η TraverseQ που σας δίνεται στο *TestQueue* θα πρέπει επίσης να τροποποιηθεί. Για να ελέγξετε την ορθότητα του προγράμματος σας

(a) Δημιουργήστε μια ουρά (QueueLimit=10) που περιλαμβάνει όλους τους ακέραιους αριθμούς στο διάστημα [0..9]. Εμφανίστε την ουρά (με βοηθητική συνάρτηση TraverseQ) την τιμή της Front, Rear και του μετρητή των στοιχείων της ουράς.

(b) Στη συνέχεια επιχειρήστε να προσθέσετε ένα οποιοδήποτε στοιχείο. Εμφανίστε την ουρά (με βοηθητική συνάρτηση TraverseQ), την τιμή της Front, Rear και του μετρητή των στοιχείων της ουράς

(c) Αφαιρέστε τη κεφαλή της ουράς και εμφανίστε την ουρά (με βοηθητική συνάρτηση TraverseQ), το στοιχείο που αφαιρέσατε, την τιμή της Front, Rear και του μετρητή των στοιχείων της ουράς

(d) Προσθέσετε ένα οποιοδήποτε στοιχείο και εμφανίστε την ουρά (με βοηθητική συνάρτηση TraverseQ), την τιμή της Front, Rear και του μετρητή των στοιχείων της ουράς

(e) Στη συνέχεια επιχειρήστε να προσθέσετε ένα οποιοδήποτε στοιχείο. Εμφανίστε την ουρά (με βοηθητική συνάρτηση TraverseQ), την τιμή της Front, Rear και του μετρητή των στοιχείων της ουράς.

(f) Αδειάστε την ουρά. Μετά την αφαίρεση κάθε φορά της κεφαλής της ουράς εμφανίστε την ουρά (με βοηθητική συνάρτηση TraverseQ), το στοιχείο που αφαιρέσατε, την τιμή της Front, Rear και του μετρητή των στοιχείων της ουράς.

```
---a---
Queue: 0 1 2 3 4 5 6 7 8 9
Front=0 Rear=0 Count=10
---b---
Full Queue
Queue: 0 1 2 3 4 5 6 7 8 9
Front=0 Rear=0 Count=10
---c---
Queue: 1 2 3 4 5 6 7 8 9
Removed item=0 Front=1 Rear=0 Count=9
---d---
Queue: 1 2 3 4 5 6 7 8 9 25
Front=1 Rear=1 Count=10
---e---
Full Queue
Queue: 1 2 3 4 5 6 7 8 9 25
Front=1 Rear=1 Count=10
---f---
Queue: 2 3 4 5 6 7 8 9 25
Removed item=1 Front=2 Rear=1 Count=9
Queue: 3 4 5 6 7 8 9 25
Removed item=2 Front=3 Rear=1 Count=8
Queue: 4 5 6 7 8 9 25
Removed item=3 Front=4 Rear=1 Count=7
Queue: 5 6 7 8 9 25
Removed item=4 Front=5 Rear=1 Count=6
Queue: 6 7 8 9 25
Removed item=5 Front=6 Rear=1 Count=5
Queue: 7 8 9 25
Removed item=6 Front=7 Rear=1 Count=4
Queue: 8 9 25
Removed item=7 Front=8 Rear=1 Count=3
Queue: 9 25
Removed item=8 Front=9 Rear=1 Count=2
Queue: 25
Removed item=9 Front=0 Rear=1 Count=1
Queue: Empty Queue
Removed item=25 Front=1 Rear=1 Count=0
Press any key to continue . . .
```

Η έξοδος του προγράμματος δίνεται στη διπλανή εικόνα.

7. Γράψτε ένα πρόγραμμα που επιτρέπει στον χρήστη να εκτελεί όλες τις βασικές λειτουργίες που σχετίζονται με τις ουρές (δες το αντίστοιχο πρόγραμμα TestQueue.c του Project_QueueADT), μέσω του παρακάτω καταλόγου επιλογών:

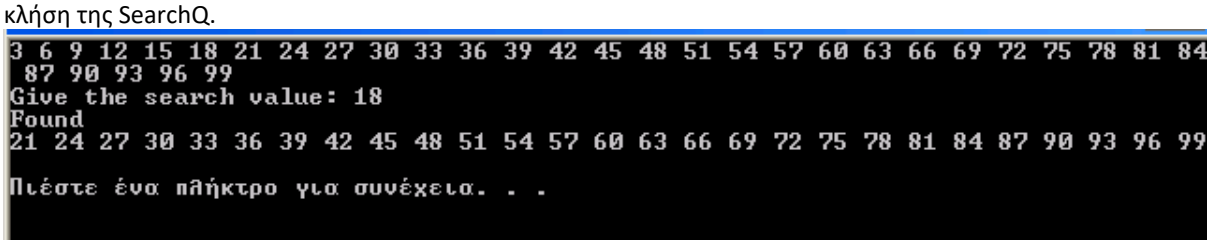
1 Δημιουργία κενής ουράς
2 Προσθήκη στοιχείου στην πίσω άκρη της ουράς
3 Εμφάνιση των στοιχείων της ουράς, χωρίς να διαγραφούν τα στοιχεία της
4 Έλεγχος κενής ουράς
5 Έλεγχος γεμάτης ουράς
6 Διαγραφή και εμφάνιση του στοιχείου της εμπρός άκρης της ουράς
7 Έξοδος από το πρόγραμμα

Για κάθε επιλογή του μενού (με εξαίρεση την τελευταία επιλογή) θα πρέπει να υλοποιήσετε μια συνάρτηση, στην οποία θα εμφανίζεται σχετικό μήνυμα για τη λειτουργία που εκτελέστηκε. Για παράδειγμα, αν ο χρήστης πληκτρολογήσει την επιλογή 2 θα εκτελεστεί η αντίστοιχη μέθοδος όπου:

- Θα εμφανιστεί το μήνυμα '*Give an item to add to the queue*'
- Ο χρήστης θα πληκτρολογήσει μία τιμή
- Θα κληθεί η συνάρτηση *AddQ*
- Αν η ουρά δεν είναι γεμάτη τότε το στοιχείο θα προστεθεί και θα εμφανιστεί το μήνυμα '*x was add at the end of the queue*', όπου *x* η τιμή που έδωσε ο χρήστης.

Στο κυρίως πρόγραμμα ο χρήστης θα πληκτρολογεί κατ' επανάληψη μία εντολή και θα καλείται η αντίστοιχη συνάρτηση μέχρι ο χρήστης να δώσει την επιλογή 2.
Τροποποιείστε κατάλληλα το TestQueue.c ώστε σε κάθε case να καλείται η αντίστοιχη συνάρτηση, δηλαδή ο κώδικας που υπάρχει σε κάθε case στο πρόγραμμα TestQueue.c θα ενσωματωθεί σε μια συνάρτηση (εκτός από την εντολή break, αυτή δε θα ενσωματωθεί μέσα στη συνάρτηση)

8. Γράψτε μια συνάρτηση *SearchQ* που δέχεται μια ουρά τύπου *QueueType* και έναν ακέραιο αριθμό *item* και εκτελεί αναζήτηση του αριθμού *item* στην ουρά (πρωτότυπο συνάρτησης `boolean SearchQ(QueueType *Queue, QueueElementType Item);`).
Η υλοποίηση της *SearchQ* θα γίνει σε επίπεδο εφαρμογής, δηλαδή με τη χρήση της *RemoveQ*. Αν το στοιχείο ***item*** υπάρχει περισσότερες από μία φορές στην ουρά τότε η αναζήτηση σταματάει όταν βρεθεί το πρώτο στοιχείο ίσο με ***item.***
Στο κυρίως πρόγραμμα θα δημιουργείτε μια ουρά που περιλαμβάνει τα πολλαπλάσια του 3 στο διάστημα [1..100], θα καλεί τη βοηθητική συνάρτηση Traverse για να διαπιστώσετε ότι η ουρά περιέχει τα παραπάνω στοιχεία. Στη συνέχεια θα δίνει ο χρήστης μία τιμή προς αναζήτηση και θα καλείται η συνάρτηση *SearchQ.* Στη συνέχεια θα καλεί τη βοηθητική συνάρτηση Traverse για να διαπιστώσετε ότι η ουρά μεταβλήθηκε μετά τη κλήση της SearchQ.



9. Στα εξωτερικά ιατρεία ενός Νοσοκομείου εκτός από τα έκτακτα περιστατικά εξυπηρετούνται τις εργάσιμες ημέρες κάποια άτομα που έχουν κλείσει ραντεβού. Συγκεκριμένα, το τηλεφωνικό κέντρο των εξωτερικών ιατρείων του Νοσοκομείου δέχεται κλήσεις από ενδιαφερόμενους που θέλουν να κλείσουν ραντεβού για την επόμενη εβδομάδα σε μία από τις 5 κλινικές που δέχονται ασθενείς με ραντεβού, βάσει του παρακάτω προγράμματος:

1) Δευτέρα	8.00-10.00	Καρδιολογικό
2) Τρίτη	8.00-10.00	Ορθοπεδικό
3)Τετάρτη	8.00-10.00	Μικροβιολογικό
4) Πέμπτη	8.00-10.00	Ακτινολογικό
5) Παρασκευή	8.00-10.00	Οφθαλμολογικό

Κάθε ραντεβού διαρκεί μία ώρα, οπότε κάθε ημέρα μπορούν να εξυπηρετηθούν 2 ασθενείς με αυστηρή σειρά προτεραιότητας (σειρά κλήσης στο τηλεφωνικό κέντρο). Στην ουσία, δηλαδή, δημιουργούνται 5 ουρές με τα ονοματεπώνυμα των ασθενών που θα εξυπηρετηθούν από κάθε κλινική την αντίστοιχη ημέρα της εβδομάδας (που καθορίζεται βάσει του παραπάνω προγράμματος). Αν ο μέγιστος αριθμός των 2 ραντεβού για

κάποια/κάποιες από τις κλινικές συμπληρωθεί τότε ο ασθενής - ανεξάρτητα από την κλινική όπου θέλει να κλείσει ραντεβού - μπαίνει σε μια άλλη ουρά εξυπηρέτησης και ενημερώνεται από το τηλεφωνικό κέντρο σε περίπτωση ακύρωσης κάποιου ραντεβού. Σε αυτή την περίπτωση εκτός από το ονοματεπώνυμο και τον αύξοντα αριθμό της κλινικής (1 = Καρδιολογικό, 2 = Ορθοπεδικό κτλ) που καταγράφει ο υπάλληλος του τηλεφωνικού κέντρου, ζητάει και το τηλέφωνο του ασθενούς προκειμένου να ειδοποιηθεί.

Να προσομοιώσετε την παραπάνω συνάρτηση χρησιμοποιώντας τον ΑΤΔ της ουράς. Συγκεκριμένα:

(a) Θα δημιουργήσετε 5 ουρές, μία για κάθε κλινική, στην οποία θα αποθηκεύονται τα ονοματεπώνυμα (*αλφαριθμητικό 25 χαρακτήρων*) των ασθενών που κλείνουν ραντεβού. Ο μέγιστος αριθμός ραντεβού και συνεπώς ασθενών για κάθε κλινική είναι 2.

(b) Θα δημιουργήσετε μία ουρά στην οποία θα αποθηκεύονται τα στοιχεία των ασθενών για τους οποίους δεν μπορούμε να κλείσουμε ραντεβού στην επιθυμητή κλινική (γιατί η αντίστοιχη ουρά είναι πλήρης). Η ουρά αυτή θα είναι κοινή για όλους τους ασθενείς που μπαίνουν στην ουρά αναμονής ανεξάρτητα από την κλινική. Ο μέγιστος αριθμός των στοιχείων αυτής της ουράς θα είναι 20 και κάθε στοιχείο της θα είναι τύπου εγγραφής με τα εξής πεδία:
   - ονοματεπώνυμο (*string 25 χαρακτήρες*)
   - κωδικός κλινικής (*int*, 1 = Καρδιολογικό, 2 = Ορθοπεδικό κτλ)
   - τηλέφωνο (*string 10 χαρακτήρες*)

(c) Θα γράψετε συνάρτηση *newAppointment* που θα δέχεται τον κωδικό μιας κλινικής, την ουρά που αντιστοιχεί στην κλινική και την ουρά αναμονής και θα ενημερώνει την κατάλληλη ουρά ζητώντας από τον χρήστη τα απαραίτητα στοιχεία: ονοματεπώνυμο στην περίπτωση που υπάρχει κενό στην επιθυμητή κλινική και ονοματεπώνυμο και τηλέφωνο στην περίπτωση που ο ασθενής πρέπει να μπει σε ουρά αναμονής. Σε κάθε περίπτωση θα ελέγχεται πρώτα αν η ουρά της αντίστοιχης κλινικής είναι γεμάτη και μετά θα προστίθεται το ραντεβού. Τέλος, η συνάρτηση θα εμφανίζει το μήνυμα '*Successful appointment for clinic x*' (όπου *x*=αύξων αριθμός κλινικής) αν το ραντεβού καταχωρηθεί στην επιθυμητή ουρά ή '*You are in a waiting list*' αν η επιθυμητή ουρά είναι γεμάτη και ο ασθενής μπει σε ουρά αναμονής.

(d) Θα γράψετε συνάρτηση *showWaitingQ* που θα δέχεται την ουρά των ασθενών που είναι σε αναμονή και θα εμφανίζει τα στοιχεία τους. (τροποποίηση της TraverseQ)

(e) Θα γράψετε συνάρτηση *showQ* που θα δέχεται την ουρά των ασθενών της κλινικής και τον κωδικό αριθμό της κλινικής θα εμφανίζει τα στοιχεία τους (τροποποίηση της TraverseQ)

Στο κυρίως πρόγραμμα, αφού δημιουργηθούν οι 6 ουρές θα καλείται κατ' επανάληψη (όπως περιγράφεται παρακάτω) η συνάρτηση *newAppointment* και θα ενημερώνεται η κατάλληλη ουρά. Ο κωδικός της κλινικής (1, 2, 3, 4 ή 5) που πρέπει να περνάει στη συνάρτηση θα παράγεται με τυχαίο τρόπο χρησιμοποιώντας τη συνάρτηση *rand (με srand(54))*. Η συνέχιση ή όχι της συνάρτησης θα γίνεται με σχετική ερώτηση '*Continue Y/N (Y=yes, N=No)?*' που θα εμφανίζεται στον χρήστη. Όταν τελειώσει η συνάρτηση καταχώρησης των ραντεβού θα καλείται η συνάρτηση *showWaitingQ* για την εμφάνιση της ουράς των ραντεβού που είναι σε αναμονή καθώς και η *showQ* για την εμφάνιση των ασθενών που έκλεισαν ραντεβού σε κάθε μια από τις 5 κλινικές. Πριν από την εμφάνιση των στοιχείων της ουράς των ασθενών που είναι σε αναμονή θα εμφανίζει το μήνυμα «Waiting list» ενώ πριν την εμφάνιση των στοιχείων κάθε ουράς που αντιστοιχεί σε μια κλινική θα εμφανίζεται το μήνυμα «Appointments of clinic x», όπου x=1, 2, 3, 4, 5 αντίστοιχα.

Υπόδειξη:

1. Θα δημιουργήσετε πίνακα 6 θέσεων (δε θα γίνει χρήση της θέσης 0 του πίνακα) για να αποθηκεύσετε τις 5 ουρές-κλινικές. Ετσι η CreateQ και showQ θα κληθούν 5 φορές μέσα από επαναληπτική συνάρτηση

2. Θα δηλώσετε 2 διαφορετικούς τύπους ουρών δηλαδή 2 διαφορετικούς τύπους QueueType, πχ QueueType1 & QueueType2 και αντίστοιχα QueueElementType1 & QueueElementType2 για να αναπαραστήσετε την ουρά-κλινική (QueueType1) και την ουρά αναμονής (QueueType2).

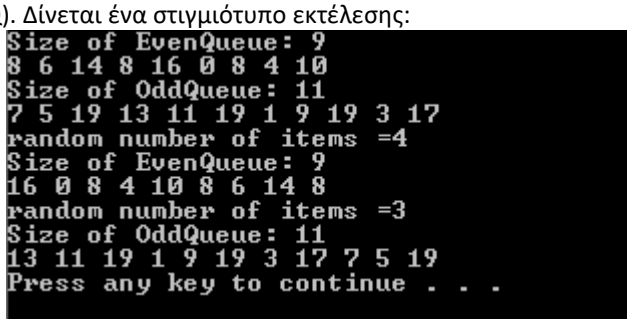
3. Εφόσον ο τύπος του στοιχείου που καταχωρείται στην ουρά είναι αλφαριθμητικό τότε θα πρέπει να γίνει η αντίστοιχη τροποποίηση σε όσες από τις λειτουργίες της ουράς είναι αναγκαίες

Στη συνέχεια δίνεται ένα **ενδεικτικό στιγμιότυπο** *από την εκτέλεση του προγράμματος.*

```
Give your name: ALPHA
Successful appointment for clinic 5
Continue Y/N (Y=yes, N=No): Y
Give your name: BETA
Successful appointment for clinic 3
Continue Y/N (Y=yes, N=No): Y
Give your name: GAMA
Successful appointment for clinic 4
Continue Y/N (Y=yes, N=No): Y
Give your name: DELTA
Successful appointment for clinic 5
```

```
Continue Y/N (Y=yes, N=No): Y
Give your name: EPSILON
Successful appointment for clinic 2
Continue Y/N (Y=yes, N=No): Y
Give your name: ZETA
Successful appointment for clinic 3
Continue Y/N (Y=yes, N=No): Y
Give your name: ETA
You are in a waiting list
Give your phone number: 123
Continue Y/N (Y=yes, N=No): Y
Give your name: THETA
You are in a waiting list
Give your phone number: 45
Continue Y/N (Y=yes, N=No): N

Appointments of clinic 1

Appointments of clinic 2
EPSILON

Appointments of clinic 3
BETA
ZETA

Appointments of clinic 4
GAMA

Appointments of clinic 5
ALPHA
DELTA

Waiting list:
ETA, 3, 123
THETA, 5, 45
```

10. Γράψτε ένα πρόγραμμα που δημιουργεί 2 ουρές μια για άρτιους και μια για περιττούς αριθμούς (QueueLimit = 20). Οι αριθμοί θα παράγονται τυχαία και αν ο τυχαίος αριθμός είναι άρτιος θα εισάγεται στην ουρά των αρτίων και αν είναι περιττός στην ουρά των περιττών ενημερώνοντας μετά από κάθε εισαγωγή το πλήθος των στοιχείων της αντίστοιχης ουράς. Θεωρείστε ότι το πλήθος των τυχαίων αριθμών είναι ίσο με QueueLimit. Στη συνέχεια, θα διαγράφεται τυχαίος αριθμός στοιχείων από κάθε ουρά και θα εισάγεται εκ νέου στην αντίστοιχη ουρά. Για τον έλεγχο της ορθότητας του κώδικα εμφανίστε το πλήθος των στοιχείων της κάθε ουράς καθώς και τα στοιχεία της ουράς (χρήση της TraverseQ). Δίνεται ένα στιγμιότυπο εκτέλεσης:



11. Γράψτε δύο συναρτήσεις minElement, maxElement, που θα δέχονται την ουρά (*με αναφορά*) και θα επιστρέφουν το μικρότερο και μεγαλύτερο στοιχείο μίας ουράς αντίστοιχα, σε επίπεδο εφαρμογής: μόνο οι λειτουργίες EmptyQ, AddQ και RemoveQ μπορούν να χρησιμοποιηθούν στην υλοποίηση των συναρτήσεων και η ουρά θα παραμένει αναλλοίωτη μετά την κλήση των συναρτήσεων. Στη συνέχεια γράψτε κυρίως πρόγραμμα, στο οποίο δημιουργείται ουρά μεγέθους 10, η οποία γεμίζει από το πληκτρολόγιο και εκτυπώνεται η ουρά, το μικρότερο και το μεγαλύτερο στοιχείο της.

12. Σε μία τράπεζα κάθε πελάτης εισέρχεται σ' αυτήν μία ορισμένη χρονική στιγμή έστω την ώρα Α και παραμένει κάποιο χρονικό διάστημα έστω Τ, προκειμένου να διεκπεραιώσει την εργασία του (ανάληψη, μεταφορά, δάνειο κλπ). Δεδομένου ότι υπάρχει μόνο ένας ταμίας, εάν κάποιος πελάτης εισέλθει ενώ δεν έχει τελειώσει ο προηγούμενος, η εξυπηρέτησή του αρχίζει αμέσως μόλις τελειώσει ο προηγούμενος. Γράψτε πρόγραμμα το οποίο μοντελοποιεί το σύστημα εξυπηρέτησης της τράπεζας, χρησιμοποιώντας τον ΑΤΔ ουρά. Κάθε πελάτης που εισέρχεται στην τράπεζα μπαίνει σε μια ουρά αναμονής. Για κάθε πελάτη μας ενδιαφέρει να γνωρίζουμε το

χρόνο άφιξης, τη διάρκεια παραμονής, την ώρα έναρξης, την ώρα λήξης της εξυπηρέτησης. Ο μέγιστος αριθμός πελατών που καταχωρούνται στην ουρά αναμονής είναι 3. Το πρόγραμμα θα διαβάζει για κάθε πελάτη την ώρα άφιξης και τον χρόνο παραμονής με τη μορφή ώραΆφιξης, χρόνοςΠαραμονής. Η ώρα άφιξης και ο χρόνος παραμονής μπορούν να είναι οποιοδήποτε θετικό μέγεθος.

Υλοποιήστε 3 συναρτήσεις:

1. Συνάρτηση που θα διαβάζει την ώρα άφιξης και τον χρόνο παραμονής του κάθε πελάτη με τη μορφή ώραΆφιξης, χρόνοςΠαραμονής και θα εισάγει τον πελάτη στην ουρά αναμονής. Για την ώρα έναρξης και ώρα λήξης θα καταχωρείται η «εικονική» τιμή -1 (η εισαγωγή των πελατών στην ουρά αναμονής θεωρήστε ότι συμβαίνει τη χρονική στιγμή που εισέρχεται στην τράπεζα). Η συνάρτηση θα είναι void με παράμετρο την ουρά αναμονής.
2. Συνάρτηση η οποία προσομοιώνει την εξυπηρέτηση των πελατών. Εξάγει έναν-έναν τον πελάτη από την ουρά αναμονής και για κάθε πελάτη που εξυπηρετείται θα υπολογίζει την ώρα έναρξης και λήξης της εξυπηρέτησης και θα καταχωρεί τον πελάτη στην ουρά εξυπηρετηθέντων (νέα ουρά). Η ώρα έναρξης της εξυπηρέτησης των πελατών είναι ο χρόνος άφιξης του πρώτου πελάτη (δες στο στιγμιότυπο εκτέλεσης). Η συνάρτηση έχει μια παράμετρο την ουρά αναμονής και επιστρέφει την ουρά των εξυπηρετηθέντων. Το πρωτότυπο της συνάρτησης είναι

   QueueType TimesInQueue(QueueType *Queue)
3. Συνάρτηση με 2 παραμέτρους: την ονομασία της ουράς, και την ουρά. Αν η ουρά δεν είναι άδεια θα εμφανίζει για κάθε πελάτη τις καταχωρημένες πληροφορίες, ενώ αν είναι άδεια θα εμφανίζει σχετικό μήνυμα (δες στο στιγμιότυπο εκτέλεσης).

   Στη συνέχεια δίνεται ένα ενδεικτικό στιγμιότυπο εκτέλεσης.

```
Give: arrival time,stay time for client 1: 9,2
Give: arrival time,stay time for client 2: 10,5
Give: arrival time,stay time for client 3: 11,4
Waiting Queue
Client                  Start   End     Arrival Stay
Client 1                -1      -1      9       2
Client 2                -1      -1      10      5
Client 3                -1      -1      11      4

Waiting Queue is empty
Service Queue
Client                  Start   End     Arrival Stay
Client 1                9       11      9       2
Client 2                11      16      10      5
Client 3                16      20      11      4
```

13. Να γράψετε πρόγραμμα που θα περιλαμβάνει μια συνάρτηση *ReverseQ*(*QueueType *Queue)* που αντιστρέφει τα στοιχεία μιας ουράς χρησιμοποιώντας τις βασικές λειτουργίες που ορίζονται στους ΑΤΔ στοίβα και ουρά (*θα χρειασθεί και στοίβα για να αντιστρέψουμε τα στοιχεία της ουράς*). Για να ελέγξετε την ορθότητα του προγράμματος σας δημιουργήστε μια ουρά που περιλαμβάνει τους όρους αριθμητικής προόδου από το 0 έως και το 10 με βήμα 0.5. Η ουρά θα εμφανιστεί πριν και μετά την αντιστροφή των στοιχείων της (κάντε χρήση της βοηθητικής συνάρτησης TraverseQ).

Στη συνέχεια δίνεται στιγμιότυπο εκτέλεσης.

```
0.00 0.50 1.00 1.50 2.00 2.50 3.00 3.50 4.00 4.50 5.00 5.50 6.00 6.50 7.00
7.50 8.00 8.50 9.00 9.50 10.00

Reverse Queue
10.00 9.50 9.00 8.50 8.00 7.50 7.00 6.50 6.00 5.50 5.00 4.50 4.00 3.50 3.00
2.50 2.00 1.50 1.00 0.50 0.00
```

14. Γράψτε μια συνάρτηση *GetNthElementValue* που δέχεται μια ουρά τύπου *QueueType* και έναν ακέραιο αριθμό *n* και επιστρέφει την τιμή του *n*-οστού στοιχείου της ουράς αφήνοντας την ουρά αναλλοίωτη. Υλοποιήστε δύο εκδόσεις της συνάρτησης:

(a) υλοποιήστε τη συνάρτηση *GetNthElementValueA* σε επίπεδο εφαρμογής: μόνο οι λειτουργίες *EmptyQ*, *AddQ* και *RemoveQ* μπορούν να χρησιμοποιηθούν.

(b) υλοποιήστε τη συνάρτηση *GetNthElementValueB* σε επίπεδο υλοποίησης: μπορείτε να προσπελάσετε άμεσα τη δομή που χρησιμοποιείται για την αποθήκευση της ουράς χωρίς να κάνετε χρήση των διαδικασιών *AddQ* και *RemoveQ*.

Στο κυρίως πρόγραμμα θα διαβάζεται το *n*, θα καλούνται οι δύο διαδικασίες και θα εμφανίζεται η τιμή που επιστρέφουν. Το *n* πρέπει να έχει τιμή μικρότερη ή ίση του πλήθους των στοιχείων της ουράς και θα ζητείται κατ' επανάληψη από τον χρήστη μέχρι να δοθεί σωστή τιμή (*n<= πλήθος στοιχείων ουράς*). Το πλήθος των στοιχείων της ουράς μπορεί να υπολογιστεί χρησιμοποιώντας τις πληροφορίες της δομής όπου αποθηκεύεται η ουρά. Για να ελέγξετε την ορθότητα των δύο διαδικασιών στο κυρίως πρόγραμμα:

1. Δημιουργήστε μια ουρά που περιλαμβάνει τα πολλαπλάσια του 5 στο διάστημα [1..100].
2. Εμφανίστε τα περιεχόμενα της ουράς.
3. Διαβάστε το n, καλέστε *GetNthElementValueA* και *GetNthElementValueB,* και εμφανίστε τις τιμές που επιστρέφουν.
4. Εμφανίστε τα περιεχόμενα της ουράς.

Ακολουθεί στιγμιότυπο εκτέλεσης:

```
5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100
Dwse n: 50
Dwse n: 6
ValueA:30
ValueB:30
5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100
```

15. Μια εναλλακτική υλοποίηση μιας ουράς που χρησιμοποιεί ένα κυκλικό πίνακα και δεν απαιτεί να διατηρούμε μία κενή θέση μεταξύ της εμπρός και της πίσω άκρη της για να ξεχωρίζει μια κενή ουρά από μια γεμάτη χρειάζεται απλά την προσθήκη ενός ακέραιου πεδίου *Count* στην εγγραφή τύπου *QueueType*, στο οποίο αποθηκεύεται ο τρέχων αριθμός στοιχείων της ουράς. Να κάνετε τις απαραίτητες αλλαγές στη δήλωση του τύπου της εγγραφής και στις βασικές λειτουργίες του ΑΤΔ ουρά με πίνακες, έτσι ώστε να χρησιμοποιείται αυτό το επιπλέον πεδίο και να μην διατηρείται κενή θέση στον πίνακα όπου αποθηκεύονται τα στοιχεία της ουράς. Η TraverseQ που σας δίνεται στο *TestQueue* θα πρέπει επίσης να τροποποιηθεί και να εμφανίζει εκτός από τα στοιχεία της ουράς, την τιμή της Front, Rear και του μετρητή των στοιχείων της ουράς. Για να ελέγξετε την ορθότητα του προγράμματος:

(a) Δημιουργήστε μια ουρά που περιλαμβάνει τα πολλαπλάσια του 3 στο διάστημα [1, 30]. Εμφανίστε την ουρά με την TraverseQ.

(b) Το πρόγραμμα θα διαβάζει έναν αριθμό, και αν είναι πολλαπλάσιο του 3 θα προσπαθεί να τον εισάγει στην ουρά. Διαφορετικά θα εμφανίζει σχετικό μήνυμα και θα διαβάζει ξανά νέο αριθμό. Εμφανίστε την ουρά με τη βοηθητική συνάρτηση TraverseQ.

(c) Αφαιρέστε τη κεφαλή της ουράς και εμφανίστε την ουρά με βοηθητική συνάρτηση TraverseQ, καθώς και το στοιχείο που αφαιρέσατε.

(d) Στη συνέχεια, θα επαναλαμβάνει δύο φορές το βήμα (b).

(e) Αδειάστε την ουρά. Μετά την αφαίρεση κάθε φορά της κεφαλής της ουράς εμφανίστε την ουρά με βοηθητική συνάρτηση TraverseQ, το στοιχείο που αφαιρέσατε, την τιμή της Front, Rear και του μετρητή των στοιχείων της ουράς.

Η έξοδος του προγράμματος δίνεται παρακάτω.

```
(a)
Queue: 3 6 9 12 15 18 21 24 27 30
Front=0 Rear=0 Count=10
(b)
Give a number:5
Give a multiple of 3
Give a number:66
Full Queue
Queue: 3 6 9 12 15 18 21 24 27 30
Front=0 Rear=0 Count=10
(c)
Queue: 6 9 12 15 18 21 24 27 30
Front=1 Rear=0 Count=9
```

```
Removed item=3
(d)
Give a number:66
Queue: 6 9 12 15 18 21 24 27 30 66
Front=1 Rear=1 Count=10
Give a number:33
Full Queue
Queue: 6 9 12 15 18 21 24 27 30 66
Front=1 Rear=1 Count=10
(e)
Queue: 9 12 15 18 21 24 27 30 66
Front=2 Rear=1 Count=9
Removed item=6
Queue: 12 15 18 21 24 27 30 66
Front=3 Rear=1 Count=8
Removed item=9
Queue: 15 18 21 24 27 30 66
Front=4 Rear=1 Count=7
Removed item=12
Queue: 18 21 24 27 30 66
Front=5 Rear=1 Count=6
Removed item=15
Queue: 21 24 27 30 66
Front=6 Rear=1 Count=5
Removed item=18
Queue: 24 27 30 66
Front=7 Rear=1 Count=4
Removed item=21
Queue: 27 30 66
Front=8 Rear=1 Count=3
Removed item=24
Queue: 30 66
Front=9 Rear=1 Count=2
Removed item=27
Queue: 66
Front=0 Rear=1 Count=1
Removed item=30
Queue: Empty Queue
Front=1 Rear=1 Count=0
Removed item=66
```