ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ

Τμήμα Εφαρμοσμένης Πληροφορικής
ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ
Εξάμηνο Β'
Φύλλο Ασκήσεων 1: ΣΥΝΟΛΑ
Μάγια Σατρατζέμη, Γεωργία Κολωνιάρη, Αλέξανδρος Καρακασίδης

Παρατηρήσεις:
1. *Τα δεδομένα εισόδου διαβάζονται 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( ) το καθένα και με τη σειρά που δηλώνονται στις εκφωνήσεις.*
3. ***ΠΡΟΣΟΧΗ: Οι ασκήσεις θα πρέπει να λύνονται με χρήση του κώδικα που υλοποιεί τον ΑΤΔ σύνολο. Ο κώδικας που σας δίνεται περιλαμβάνεται στο code.zip στην αντίστοιχη διάλεξη.***

1. Ένα σημαντικό πρόβλημα της *θεωρίας αριθμών* (του πεδίου των μαθηματικών που έχει ως αντικείμενο τη μελέτη των ιδιοτήτων των μη αρνητικών ακεραίων) είναι το πρόβλημα του προσδιορισμού του κατά πόσο ένας ακέραιος είναι *πρώτος*. Ένας θετικός ακέραιος *n* είναι **πρώτος** (prime) αν έχει ακριβώς δύο θετικούς διαιρέτες, τον εαυτό του και το 1. Οι πρώτοι αριθμοί είναι ιδιαίτερα σημαντικοί σήμερα για την *κρυπτογραφία*, αφού πολλές από τις καλύτερες τεχνικές κρυπτογράφησης βασίζονται στους πρώτους αριθμούς. Να γραφεί πρόγραμμα που θα περιλαμβάνει τα παρακάτω υποπρογράμματα:
- συνάρτηση *isPrime* που θα δέχεται ένα θετικό ακέραιο *n* και θα επιστρέφει την τιμή *TRUE* ή *FALSE* ανάλογα με το αν ο *n* είναι ή όχι αντίστοιχα πρώτος αριθμός
- διαδικασία *createPrimeSet* που θα δέχεται δύο θετικούς ακεραίους, έστω *first* και *last*, και θα δημιουργεί και επιστρέφει το σύνολο των πρώτων αριθμών που ανήκουν στο διάστημα [*first* .. *last*]
- διαδικασία *displaySet* που δέχεται ένα σύνολο θετικών ακεραίων *S*, τον πρώτο (*first*) και τον τελευταίο (*last*) αριθμό αυτού του συνόλου και εμφανίζει τα στοιχεία του συνόλου στην ίδια γραμμή με ένα κενό χαρακτήρα μεταξύ τους.

Στη συνέχεια, γράψτε κυρίως πρόγραμμα όπου θα διαβάζονται δύο ακέραιοι αριθμοί, έστω *first* και *last*, που θα πρέπει να ανήκουν στο διάστημα [2..200] και επιπλέον να ισχύει *first* ≤ *last*. Στη συνέχεια, θα καλούνται οι διαδικασίες *createPrimeSet* και *displaySet* για τη δημιουργία και εμφάνιση του συνόλου των πρώτων αριθμών που ανήκουν στο διάστημα [*first* .. *last*].
Να χρησιμοποιηθεί η υλοποίηση του ΑΤΔ σύνολο με πίνακα.

2. Ένας *μεταγλωττιστής* (compiler) είναι ένα πρόγραμμα που μεταφράζει ένα πηγαίο πρόγραμμα που έχει γραφεί σε κάποια γλώσσα υψηλού επιπέδου, όπως η C, σε ένα αντικείμενο πρόγραμμα σε γλώσσα μηχανής. Ένα βασικό τμήμα ενός μεταγλωττιστή είναι ο *λεκτικός αναλυτής* (lexical analyzer) που αναγνωρίζει τις *λεκτικές μονάδες* (tokens) του πηγαίου προγράμματος. Ένας λεκτικός αναλυτής περιλαμβάνει συναρτήσεις που ελέγχουν αν μια ακολουθία χαρακτήρων αποτελεί αναγνωριστικό, δεσμευμένη λέξη, ακέραιο αριθμό κτλ. Να γράψετε πρόγραμμα που θα περιλαμβάνει συναρτήσεις με ονόματα:
- *isValidInteger*
- *isValidIdentifier*

που θα δέχονται ένα αλφαριθμητικό και θα επιστρέφουν την τιμή *TRUE* ή *FALSE* ανάλογα με το αν αυτό αποτελεί *ακέραιο* αριθμό της C ή *αναγνωριστικό* (identifier) αντίστοιχα. Θυμηθείτε ότι:
- Ένας ακέραιος αριθμός στην C μπορεί να ξεκινάει με '+' ή '-' και περιλαμβάνει μόνο τα ψηφία 0..9.
- Ένα αναγνωριστικό μπορεί να περιλαμβάνει γράμματα (κεφαλαία και πεζά) του λατινικού αλφαβήτου, αριθμητικά ψηφία και τον χαρακτήρα υπογράμμισης. Ένα αναγνωριστικό αρχίζει υποχρεωτικά από γράμμα και δεν περιέχει κανένα κενό. Το γεγονός ότι δεν μπορεί να χρησιμοποιηθεί δεσμευμένη λέξη ως αναγνωριστικό δεν θα ληφθεί υπόψη σε αυτή την άσκηση.

Στο κυρίως πρόγραμμα θα διαβάζεται κατ' επανάληψη ένα αλφαριθμητικό και θα ελέγχεται αν αυτό αποτελεί έγκυρο ακέραιο αριθμό ή όχι, οπότε θα εμφανίζεται το μήνυμα *'Valid integer'* ή *'Not a valid integer'* αντίστοιχα. Στη συνέχεια, θα διαβάζεται κατ' επανάληψη ένα αλφαριθμητικό και θα ελέγχεται αν αυτό αποτελεί έγκυρο αναγνωριστικό ή όχι, οπότε θα εμφανίζεται το μήνυμα *'Valid identifier'* ή *'Not a valid identifier'* αντίστοιχα. Και στις δυο περιπτώσεις η συνέχιση ή όχι της εισαγωγής αλφαριθμητικών θα ελέγχεται από σχετικό μήνυμα στο οποίο ο χρήστης θα απαντάει εισάγοντας ένα χαρακτήρα από τους: 'Y', 'y', 'N', 'n', όπου: 'Y', 'y' = Yes, 'N', 'n' = No.

Να χρησιμοποιηθεί ο ΑΤΔ σύνολο με πίνακα και να δημιουργήσετε τα σύνολα

```
CharacterSet : '+', '-'
DigitSet : '0' έως '9'
LetterSet : 'A' έως 'Z', 'a' έως 'z'
```

Τα παραπάνω σύνολα θα πρέπει να δημιουργηθούν χρησιμοποιώντας τη διαδικασία `Eisagogi` που υλοποιείται στον ΑΤΔ σύνολο με πίνακα. Ως καθολικό σύνολο θεωρήστε το σύνολο των χαρακτήρων του κώδικα ASCII (για την αναπαράσταση του μπορείτε να χρησιμοποιήσετε ένα πίνακα [0..255]).

3. Χρησιμοποιώντας την υλοποίηση του ΑΤΔ σύνολο με πίνακα και θεωρώντας το καθολικό σύνολο των κεφαλαίων γραμμάτων του αγγλικού αλφαβήτου να γράψετε πρόγραμμα όπου:

(a) θα δημιουργείται και θα εμφανίζεται το καθολικό σύνολο.
(b) θα δημιουργούνται και θα εμφανίζονται τα σύνολα *S ={A, B, C, D}* και *T = {A, C, E, G, I}*.
(c) θα ελέγχεται αν τα σύνολα *S* και *T* είναι ίσα, αν το *S* είναι υποσύνολο του *T*, καθώς επίσης και αν το *T* είναι υποσύνολο του *S*. Αν αληθεύει κάποιος από τους παραπάνω ελέγχους θα εμφανίζεται το μήνυμα *'ISA SYNOLA'*, *'S YPOSYNOLO T'* ή *'T YPOSYNOLO S'* αντίστοιχα, διαφορετικά δεν θα εμφανίζεται τίποτα.
(d) θα υπολογίζεται και θα εμφανίζεται η ένωση των συνόλων *S* και *T*.
(e) θα υπολογίζεται και θα εμφανίζεται η τομή των συνόλων *S* και *T*.
(f) θα υπολογίζεται και θα εμφανίζεται η διαφορά των συνόλων *S - T*.

Για την εμφάνιση ενός συνόλου θα πρέπει να υλοποιήσετε μια διαδικασία *EmfanisiSynolou* που θα δέχεται το σύνολο και θα το εμφανίζει. Κατά την εμφάνιση ενός συνόλου θα εμφανίζονται οι χαρακτήρες του αγγλικού αλφαβήτου και όχι οι λογικές τιμές *TRUE* και *FALSE* που υπάρχουν στην αντίστοιχη θέση του πίνακα που χρησιμοποιείται για την αναπαράσταση του. Υπόδειξη: ένας εύκολος τρόπος υλοποίησης της διαδικασίας είναι η δήλωση ενός αλφαριθμητικού με τιμή *'ABCDEFGHIJKLMNOPQRSTUVWXYZ'* και η εμφάνιση εκείνων των χαρακτήρων του αλφαριθμητικού για τους οποίους υπάρχει στις θέσεις που τους έχουν αντιστοιχηθεί στον πίνακα η τιμή *TRUE*.

4. Χρησιμοποιώντας την υλοποίηση του ΑΤΔ σύνολο με πίνακα και θεωρώντας το καθολικό σύνολο με τύπο βάσης 1..59 να γράψετε πρόγραμμα όπου:

(a) θα δημιουργείται και θα εμφανίζεται το σύνολο των περιττών αριθμών.
(b) θα δημιουργείται και θα εμφανίζεται το σύνολο των πρώτων αριθμών.
(c) θα υπολογίζεται και θα εμφανίζεται η τομή του συνόλου των περιττών και των πρώτων αριθμών.
(d) θα υπολογίζεται και θα εμφανίζεται η ένωση του συνόλου των περιττών και των πρώτων αριθμών.
(e) θα υπολογίζεται και θα εμφανίζεται το σύνολο των περιττών αριθμών που δεν είναι πρώτοι αριθμοί.
(f) θα υπολογίζεται και θα εμφανίζεται το συμπλήρωμα του συνόλου των πρώτων αριθμών.

Στην υλοποίηση του ΑΤΔ σύνολο με πίνακα θα πρέπει να συμπληρώσετε τα εξής υποπρογράμματα:

- συνάρτηση *isPrime* που θα δέχεται ένα θετικό ακέραιο *n* και θα επιστρέφει την τιμή *TRUE* ή *FALSE* ανάλογα με το αν ο *n* είναι ή όχι αντίστοιχα πρώτος αριθμός. Ένας θετικός ακέραιος *n* είναι **πρώτος** (prime) αν έχει ακριβώς δύο θετικούς διαιρέτες, τον εαυτό του και το 1.
- διαδικασία *ComplementSynolou* που θα δέχεται ένα σύνολο *S* και θα επιστρέφει το συμπλήρωμα του. Το συμπλήρωμα ενός συνόλου *S* περιλαμβάνει όλα που ανήκουν στο καθολικό σύνολο και δεν ανήκουν στο σύνολο *S*.
- Για την εμφάνιση των συνόλων χρησιμοποιείστε τη διαδικασία *displaySet (από το TestSetADT.c)*.

5. Στα μαθηματικά, η ακολουθία Fibonacci ορίζεται ως το σύνολο των αριθμών που προκύπτουν από το άθροισμα των δύο προηγούμενων αριθμών του συνόλου. Εξ ορισμού, οι 2 πρώτοι αριθμοί του συνόλου είναι οι 0,1. Να γραφεί πρόγραμμα το οποίο θα υλοποιεί τις παρακάτω συναρτήσεις:

a. Συνάρτηση isFibonacci, η οποία δέχεται έναν θετικό ακέραιο n και μία ακολουθία Fibonacci (typos_synolou) και επιστρέφει την τιμή TRUE ή FALSE ανάλογα εάν ο αριθμός ανήκει ή όχι αντίστοιχα στην ακολουθία Fibonacci
b. Διαδικασία createFibonacciSet η οποία θα δέχεται έναν θετικό ακέραιο threshold και θα δημιουργεί και επιστρέφει το σύνολο Fibonacci, μέχρι και τον αριθμό που είναι μικρότερος ή ίσος από τον δοσμένο ακέραιο.

Στη συνέχεια, γράψτε κυρίως πρόγραμμα το οποίο θα ζητάει από τον χρήστη έναν ακέραιο αριθμό ***max***, ο οποίος ανήκει στο διάστημα [2…1000] και θα δημιουργεί και θα εμφανίζει την ακολουθία Fibonacci, όπου το μεγαλύτερο στοιχείο της θα είναι μικρότερο ή ίσο του *max*. Χρησιμοποιείστε την υλοποίηση ΑΔΤ σύνολο με πίνακα και τη διαδικασία displaySet από το TestSetADT.c για την εμφάνιση του συνόλου. Τέλος, μετά την εμφάνιση του συνόλου, ο χρήστης θα μπορεί να εισάγει αριθμούς επανηλητικά, τους

οποίους το πρόγραμμα θα ελέγχει για το εάν ανήκουν στην τρέχουσα ακολουθία Fibonacci και θα εκτυπώνει αντίστοιχο μήνυμα Το πρόγραμμα θα τερματίζει όταν λάβει αρνητικό αριθμό. ***Στη συνέχεια δίνονται 2 στιγμιότυπα εκτέλεσης:***

```
Dwse to megistoarithmo. 200
 0 1 2 3 5 8 13 21 34 55 89 144
Enter number to check: 5
Fibonacci!
Enter number to check: 6
Not Fibonacci...
Enter number to check: 144
Fibonacci!
Enter number to check: 233
Not Fibonacci...
Enter number to check: 377
Not Fibonacci...
Enter number to check: -1
Press any key to continue . . .
```

```
Dwse to megistoarithmo. 1000
 0 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
Enter number to check: 144
Fibonacci!
Enter number to check: -1
Press any key to continue . . .
```

6. Το αγγλικό αλφάβητο αποτελείται από 26 γράμματα, εκ των οποίων τα 6 είναι φωνήεντα : A,E,I,O,U,Y. Τα υπόλοιπα είναι σύμφωνα. Να γραφεί πρόγραμμα το οποίο θα υλοποιεί τις παρακάτω συναρτήσεις:
   a. Συνάρτηση *createAlphabetSet* η οποία δημιουργεί το σύνολο των **κεφαλαίων** Αγγλικών γραμμάτων.
   b. Συνάρτηση *isVowel*, δέχεται έναν χαρακτήρα *c* και το σύνολο των φωνηέντων και επιστρέφει την τιμή *TRUE* ή *FALSE* ανάλογα εάν είναι φωνήεν ή όχι.
   c. Συνάρτηση *createVowelSet,* η οποία δημιουργεί και επιστρέφει το σύνολο των φωνηέντων.
   d. Συνάρτηση *vowelCount,* η οποία δέχεται έναν πίνακα χαρακτήρων και το σύνολο των φωνηέντων και επιστρέφει τον αριθμό των φωνηέντων στον πίνακα.

Στη συνέχεια γράψτε κυρίως πρόγραμμα το οποίο θα δέχεται από τον χρήστη μία συμβολοσειρά **κεφαλαίων Αγγλικών γραμμάτων** μήκους το πολύ 16 χαρακτήρων και θα εμφανίζει τον αριθμό των φωνηέντων σε αυτήν. Σε περίπτωση που η συμβολοσειρά περιέχει ακατάλληλους χαρακτήρες (δηλαδή χαρακτήρες που δεν ανήκουν στο σύνολο των Αγγλικών κεφαλαίων) θα εμφανίζεται σχετικό μήνυμα για κάθε άκυρο χαρακτήρα και στη συνέχεια και πάλι το πλήθος των φωνηέντων. ***Στη συνέχεια δίνετε 1 στιγμιότυπο εκτέλεσης***

```
Parakalw eisagete mia sumvoloseira KEFALAIWN TO POLY 16 XARAKTHRWN. GOODMORNING
Number of vowels: 4
Parakalw eisagete mia sumvoloseira KEFALAIWN TO POLY 16 XARAKTHRWN.TO BE OR NOT TO BE
Invalid character detected: 32 .
Invalid character detected: 32 .
Invalid character detected: 32 .
Invalid character detected: 32 .
Invalid character detected: 32 .
Number of vowels: 5
Parakalw eisagete mia sumvoloseira KEFALAIWN TO POLY 16 XARAKTHRWN.
```

7. Στα μαθηματικά, το δυναμοσύνολο ενός συνόλου A είναι το σύνολο εκείνο το οποίο περιέχει όλα τα δυνατά υποσύνολα του A. Για παράδειγμα το δυναμοσύνολο του {1,2} είναι το σύνολο {{},{1},{2},{1,2}}. Εάν το σύνολο A έχει πληθικό αριθμό N, τότε το δυναμοσύνολό του αποτελείται από 2^N σύνολα. Παρατηρείστε ότι εάν το αρχικό σύνολο μπορεί να αναπαρασταθεί από N bits (έστω N= 4, της μορφής 0111,0011,0101 κλπ..) το δυναμοσύνολό του είναι το σύνολο των αριθμών {0…2^{N-1}} σε δυαδική μορφή. Χρησιμοποιώντας αυτή τη γνώση, και τους τελεστές της C:
   a. [<<](bit-wise μετατόπιση. 1<<N σημαίνει μετατόπιση του 1 κατά N bits. Στην ουσία παραγωγή ενός integer με μόνο ένα bit μονάδα στην θέση N)
   b. [&] (bit-wise AND. a & b = 1 μόνο εάν a=1 ΚΑΙ b=1)

Γράψτε μία συνάρτηση createPowerSet η οποία παράγει το δυναμοσύνολο του καθολικού συνόλου και το επιστρέφει.

Στη συνέχεια γράψτε κυρίως πρόγραμμα το οποίο δημιουργεί το δυναμοσύνολο του καθολικού συνόλου και το εμφανίζει για N=5. ***Στη συνέχεια δίνετε 1 στιγμιότυπο εκτέλεσης***

```
TO DYNAMOSYNOLO GIA N=5

 1
 2
 1 2
 3
 1 3
 2 3
 1 2 3
 4
 1 4
 2 4
 1 2 4
 3 4
 1 3 4
 2 3 4
 1 2 3 4
 5
 1 5
 2 5
 1 2 5
 3 5
 1 3 5
 2 3 5
 1 2 3 5
 4 5
 1 4 5
 2 4 5
 1 2 4 5
 3 4 5
 1 3 4 5
 2 3 4 5
 1 2 3 4 5
Press any key to continue . . .
```