ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ (2011-12)

Άσκηση 4

Στην άσκηση αυτή, καλείσθε να υλοποιήσετε σε C ένα *διερμηνέα* (interpreter) για μία γλώσσα προγραμματισμού που αποτελεί, σε γενικές γραμμές, ένα μικρό υποσύνολο της γλώσσας C. Ας ονομάσουμε αυτή τη γλώσσα IPL (**I**ntroduction to **P**rogramming **L**anguage). Διερμηνέας για μία γλώσσα προγραμματισμού είναι ένα πρόγραμμα το οποίο εκτελεί τις εντολές ενός προγράμματος της γλώσσας, μία-μία σε πηγαίο επίπεδο, σύμφωνα με τη λειτουργικότητα της κάθε εντολής, όπως αυτή ορίζεται στη γλώσσα.[1]

Προδιαγραφές της IPL

Αρχικά, περιγράφονται οι προδιαγραφές της γλώσσας. Κάποιες από αυτές είναι εκτός του βασικού τμήματος της άσκησης και στην περίπτωση που υλοποιηθούν, θα υπάρξει επιπλέον βαθμολογική ανταπόδοση (bonus), μέχρι ένα μέγιστο ποσοστό που καθορίζεται ρητά. Όπου δεν αναφέρεται bonus βαθμολογία, εννοείται ότι η σχετική λειτουργικότητα εντάσσεται μέσα στο βασικό τμήμα της άσκησης.

Σταθερές: Η IPL υποστηρίζει, ως σταθερές, μόνο μη αρνητικούς ακέραιους αριθμούς, στο επίπεδο του πηγαίου προγράμματος. Όμως, ενδιάμεσα αποτελέσματα που προκύπτουν από αριθμητικές πράξεις, ή αριθμοί που διαβάζονται από την είσοδο ή εκτυπώνονται στην έξοδο, μπορεί να είναι και αρνητικοί ακέραιοι.

Μεταβλητές: Η IPL υποστηρίζει μόνο ακέραιες μεταβλητές. Γι΄ αυτό το λόγο δεν υπάρχει ανάγκη να δηλώνονται ρητά στην αρχή του προγράμματος, όπως γίνεται στη C. Τα ονόματα των μεταβλητών μπορούν να περιέχουν πεζούς ή κεφαλαίους λατινικούς χαρακτήρες, ψηφία και τον χαρακτήρα _ (underscore), με τον πρώτο χαρακτήρα να είναι πεζός ή κεφαλαίος χαρακτήρας. Για παράδειγμα, αποδεκτά ονόματα μεταβλητών είναι τα `x`, `A_divisor`, `count27`, ενώ μη αποδεκτά είναι τα `33abc`, `w$x76`, `_qwerty`. Όλες οι μεταβλητές ενός προγράμματος IPL είναι αρχικοποιημένες στην τιμή 0.

Είσοδος/έξοδος: Η εντολή εισόδου της IPL είναι η

```
read <var>
```

όπου `<var>` είναι το όνομα μίας μεταβλητής. Με την εντολή διαβάζεται από την είσοδο ένας ακέραιος αριθμός και ανατίθεται ως τιμή στη μεταβλητή `<var>`.

Η IPL έχει δύο εντολές εξόδου, τις

```
write <varint>
```

και

```
writeln <varint>
```

όπου `<varint>` είναι όνομα μεταβλητής ή μη αρνητικός ακέραιος. Και οι δύο εντολές εκτυπώνουν στην έξοδο την τιμή της μεταβλητής ή τον ακέραιο αριθμό. Η `write` εκτυπώνει αμέσως μετά και ένα κενό χαρακτήρα, ενώ η `writeln` εκτυπώνει αμέσως μετά αλλαγή γραμμής.

[1] Θερμές ευχαριστίες στον Ορέστη Πολυχρονίου, απόφοιτο του προπτυχιακού και μεταπτυχιακού προγράμματος σπουδών του Τμήματος, τώρα διδακτορικό φοιτητή στο Πανεπιστήμιο Columbia στη Νέα Υόρκη, για την ιδέα αυτής της άσκησης, τον σχεδιασμό της IPL και την υλοποίηση ενός διερμηνέα και ενός μεταγλωττιστή για τη γλώσσα αυτή.

Αριθμητικές παραστάσεις: Στην IPL υποστηρίζονται αριθμητικές παραστάσεις μεταξύ δύο μόνο μεταβλητών ή μη αρνητικών ακεραίων. Δηλαδή είναι της μορφής

`<varint1> <operator> <varint2>`

όπου τα `<varint1>` και `<varint2>` είναι μεταβλητές ή μη αρνητικοί ακέραιοι και το `<operator>` μπορεί να είναι κάποιος τελεστής από τους `+`, `-`, `*`, `/`, `%`, με την ίδια σημασία που έχουν οι τελεστές αυτοί στη C, για την ακέραια αριθμητική.

Εντολή αντικατάστασης: Η εντολή αντικατάστασης της IPL είναι (περίπου) όπως και στη C:

`<var> = <varint>`

ή

`<var> = <expression>`

Στην πρώτη μορφή, το `<varint>` είναι μία μεταβλητή ή ένας μη αρνητικός ακέραιος, οπότε η αντίστοιχη τιμή ανατίθεται στη μεταβλητή `<var>`. Στη δεύτερη μορφή, η τιμή της αριθμητικής παράστασης `<expression>` ανατίθεται στη μεταβλητή `<var>`.

Συνθήκες: Η IPL υποστηρίζει συνθήκες που μπορεί να είναι αριθμητικές συγκρίσεις μεταξύ μεταβλητών ή μη αρνητικών ακεραίων. Είναι της μορφής:

`<varint1> <compoper> <varint2>`

όπου τα `<varint1>` και `<varint2>` είναι μεταβλητές ή μη αρνητικοί ακέραιοι και το `<compoper>` μπορεί να είναι κάποιος τελεστής από τους `==`, `!=`, `<=`, `<`, `>=`, `>` με την ίδια σημασία που έχουν οι τελεστές αυτοί στη C.

Εντολή `while`: Η IPL υποστηρίζει σαν δομή επανάληψης μόνο την `while`. Για να επισημανθεί ποιες εντολές βρίσκονται μέσα στο σώμα μίας εντολής `while`, οι γραμμές των εντολών αυτών πρέπει να έχουν στην αρχή τους ένα στηλογνώμονα (tab). Για παράδειγμα:

```
while <condition>
< t a b ><statement1>
< t a b ><statement2>
< t a b ><statement3>
..........
```

όπου το `<condition>` είναι μία συνθήκη, οι εντολές `<statement1>`, `<statement2>`, ... βρίσκονται μέσα στο σώμα του `while` και το `< t a b >` παριστάνει τον χαρακτήρα στηλογνώμονα. Υποστηρίζονται και εμφωλευμένες `while`, σε οποιοδήποτε βάθος, απλώς πρέπει στις γραμμές των εντολών μέσα στα σώματα των πιο εσωτερικών `while` να υπάρχει ο κατάλληλος αριθμός στηλογνωμόνων στην αρχή, ώστε να είναι σαφές κάθε εντολή μέσα σε ποια `while` ανήκει. Ένα παράδειγμα εμφωλευμένων `while`, στο οποίο οι στηλογνώμονες στην αρχή των γραμμών φαίνονται σαν κενά διαστήματα και είναι ένας, δύο ή τρεις, ανάλογα με το βάθος της εμφώλευσης, είναι το εξής:

```
<statement01>
while <condition1>
        <statement11>
        <statement12>
        while <condition2>
                <statement21>
        <statement13>
        while <condition3>
                while <condition4>
                        <statement41>
                <statement31>
                <statement32>
        <statement14>
<statement02>
```

Εντολή `if/else`**:** Στην IPL υποστηρίζεται η εντολή `if` με προαιρετικό και ένα τμήμα `else`. Η επισήμανση για το ποιες εντολές περιέχονται στο τμήμα της `if` που θα εκτελεσθούν αν ισχύει η συνθήκη και ποιες στο τμήμα `else` γίνεται, όπως και στην εντολή `while`, μέσω στηλογνωμόνων στην αρχή των γραμμών. Επιτρέπονται και εμφωλευμένες `if` μέσα σε `if`, όπως και εμφωλευμένες `while` μέσα σε `if` και `if` μέσα σε `while`. Ένα παράδειγμα:

```
if <condition1>
        <statement11a>
        <statement12a>
else
        <statement11b>
        if <condition2>
                <statement21a>
                if <condition3>
                        <statement31a>
                else
                        <statement31b>
                <statement22a>
        <statement12b>
```

Τυχαίοι αριθμοί: Με την εντολή

```
random <var>
```

ανατίθεται στη μεταβλητή `<var>` σαν τιμή ένας τυχαίος μη αρνητικός ακέραιος.

Σχόλια: Τα προγράμματα IPL μπορούν να περιέχουν και σχόλια. Σε κάθε γραμμή του προγράμματος, οτιδήποτε βρίσκεται μετά από τον χαρακτήρα `#` θεωρείται γραμμή σχολίου. Πριν το `#`, μπορεί να υπάρχει κανονικά εντολή του προγράμματος. Επίσης, οι κενές γραμμές στο πηγαίο πρόγραμμα ή γραμμές μόνο με κενούς χαρακτήρες και στηλογνώμονες αγνοούνται.

Ορίσματα γραμμής εντολής: Τα προγράμματα IPL μπορούν να δέχονται κατά την εκτέλεσή τους και ορίσματα γραμμής εντολής. Με την εντολή

```
argument size <var>
```

ανατίθεται στη μεταβλητή <var> σαν τιμή το πλήθος των ορισμάτων που δόθηκαν κατά την κλήση του προγράμματος. Δεν προσμετράται στα ορίσματα το όνομα του προγράμματος. Με την εντολή

```
argument <varint> <var>
```

όπου <varint> είναι μεταβλητή ή ακέραιος μεγαλύτερος ή ίσος του 1, ανατίθεται στη μεταβλητή <var> σαν τιμή το <varint>-οστό όρισμα που δόθηκε κατά την κλήση του προγράμματος.

Εντολές `break` και `continue` (bonus 10% - 15%): Η IPL υποστηρίζει τις εντολές `break` και `continue` μέσα σε δομές επανάληψης `while` με ακριβώς την ίδια λειτουργικότητα που έχουν οι εντολές αυτές στη C. Επίσης, υποστηρίζεται και μία επέκταση των εντολών αυτών. Συγκεκριμένα, οι εντολές

```
break <n>
```

και

```
continue <n>
```

όπου το <n> είναι θετικός ακέραιος, κάνουν `break` ή `continue`, αντίστοιχα, στο <n>-οστό εξωτερικό `while` σε σχέση με αυτό στο οποίο βρίσκονται. Οι εντολές `break 1` και `continue 1` είναι ισοδύναμες με τις `break` και `continue`, αντίστοιχα.

Η υποστήριξη από την IPL μόνο των εντολών `break` και `continue` αντιστοιχεί σε bonus 10%, ενώ αν υποστηρίζονται και οι εκτεταμένες εκδοχές τους (με το <n>), το bonus μπορεί να φτάσει το 15%.

Πίνακες (bonus 30% - 35%): Η IPL υποστηρίζει μονοδιάστατους πίνακες ακεραίων. Ένας πίνακας ορίζεται και δεσμεύεται μνήμη γι' αυτόν με την εντολή

```
new <name>[<varintnozero>]
```

όπου <name> είναι το όνομα ενός πίνακα και <varintnozero> είναι η διάσταση του πίνακα, σαν η τιμή μίας μεταβλητής ή σαν ένας θετικός ακέραιος. Δεν επιτρέπεται να οριστεί πίνακας με διάσταση 0 ή αρνητική. Δεν επιτρέπεται το όνομα ενός πίνακα να συμπίπτει με το όνομα μίας μεταβλητής του προγράμματος. Τα στοιχεία ενός πίνακα είναι αρχικοποιημένα στην τιμή 0.

Η αποδέσμευση μνήμης για έναν πίνακα γίνεται με την εντολή

```
free <name>
```

όπου <name> είναι το όνομα του πίνακα.

Η αναφορά στα στοιχεία ενός πίνακα γίνεται, όπως και στη C, με την έκφραση <name>[<i>], όπου <i> είναι μεταβλητή ή ακέραιος που παριστάνει το δείκτη του στοιχείου του πίνακα. Το <i> οφείλει να είναι από 0 έως τη διάσταση του πίνακα μείον 1. Αν βρίσκεται έξω από αυτά τα όρια, η IPL δίνει λάθος εκτέλεσης.

Με την εντολή

`size <name> <var>`

δίνεται στη μεταβλητή `<var>` σαν τιμή η διάσταση του πίνακα `<name>`.

Αν κατά την αναφορά στα στοιχεία ενός πίνακα, με την έκφραση `<name>[<i>]`, επιτρέπεται το `<i>` να είναι, εκτός από μεταβλητή ή ακέραιος, και στοιχείο πίνακα, το bonus για την υλοποίηση και αυτής της λειτουργικότητας αντιστοιχεί σε 5% (επιπλέον του βασικού 30% για την υποστήριξη των πινάκων). Για παράδειγμα, η επιπλέον αυτή λειτουργικότητα δίνει τη δυνατότητα χρησιμοποίησης εκφράσεων σαν την `x[y[z[w[i]]]]`.

Ενδεικτικά IPL προγράμματα

Στη συνέχεια δίνονται κάποια ενδεικτικά IPL προγράμματα. Τα αρχεία πηγαίων προγραμμάτων της IPL θεωρούνται ότι έχουν επέκταση "`.ipl`". Όλα τα αρχεία αυτά είναι προσπελάσιμα κάτω από το `http://www.di.uoa.gr/~ip/hwfiles/ipli`.

Πρόγραμμα `primes.ipl`: Να βρεθούν όλοι οι πρώτοι αριθμοί που είναι μικρότεροι η ίσοι από αριθμό που διαβάζεται από την είσοδο.

```
# Filename: primes.ipl
#
# Find all prime numbers less than or equal to a given number

read maxnumb
number = 1
while number <= maxnumb
        if number == 1
                is_prime = 0
        else
                is_prime = 1
        divisor = 2
        if number < 4
                stop = 1
        else
                stop = 0
        while stop == 0
                remainder = number % divisor
                if remainder == 0
                        is_prime = 0
                divisor = divisor + 1
                if is_prime == 0
                        stop = 1
                divisor2 = divisor * divisor
                if divisor2 > number
                        stop = 1
        if is_prime != 0
                writeln number
        number = number + 1
```

Πρόγραμμα `happypr.ipl`: Να γεννηθούν 100000 τυχαία ζευγάρια αριθμών y και z και να βρεθεί ποιοι αριθμοί x που δίνονται από τον τύπο $x = ((y \bmod 32768) + 1) \cdot ((z \bmod 32768) + 1) + 1$ είναι ευτυχισμένοι πρώτοι (άσκηση 1), όπου το mod συμβολίζει το υπόλοιπο διαίρεσης.

```
# Filename: happypr.ipl
#
# Generate 100000 pairs of random numbers y, z
# and find the numbers x produced by the formula
# x = ((y mod 32768) + 1) * ((z mod 32768) + ) + 1
# which are happy primes (cf. homework 1)

iteration = 1
while iteration <= 100000
        # generate random value
        random y
        y = y % 32768
        y = y + 1
        random z
        z = z % 32768
        z = 1 + z
        x = y * z
        x = x + 1
        # check if prime
        is_prime = 1
        divisor = 3
        stop_check = 0
        if x < 4
                stop_check = 1
        else
                # check division with two
                remainder = x % 2
                if remainder == 0
                        stop_check = 1
                        is_prime = 0
        # check division with rest odd divisors
        while stop_check == 0
                remainder = x % divisor
                if remainder == 0
                        is_prime = 0
                divisor = divisor + 2
                if is_prime == 0
                        stop_check = 1
                divisor2 = divisor * divisor
                if divisor2 > x
                        stop_check = 1
        if is_prime != 0
                # check if happy
                stop_check = 0
                sequence = x
                # compute the sequence
                while stop_check == 0
                        # sum digits
                        sum = 0
                        while sequence != 0
                                digit = sequence % 10
                                digit = digit * digit
                                sum = sum + digit
                                sequence = sequence / 10
                        # happy or infinite loop
                        if sum == 1
                                is_happy = 1
                                stop_check = 1
                        else
                                if sum == 89
                                        is_happy = 0
                                        stop_check = 1
                        sequence = sum
                # print if happy
                if is_happy != 0
                        writeln x
        iteration = iteration + 1
```

Πρόγραμμα `selectsort.ipl`: Να ταξινομηθεί σε αύξουσα σειρά ένας πίνακας ακεραίων με τη μέθοδο της επιλογής (selection sort). Το μέγεθος του πίνακα να δίνεται σαν όρισμα από τη γραμμή εντολής (αν δεν δοθεί όρισμα, το μέγεθος να ισούται με 1000). Τα στοιχεία του πίνακα παράγονται τυχαία με τον εξής τρόπο. Αν είναι n η διάσταση του πίνακα, κάθε στοιχείο του είναι της μορφής[2] $x \bmod ((9 * n) \operatorname{div} 10) + (n \operatorname{div} 10)$, όπου το x είναι τυχαίος αριθμός και τα mod/div συμβολίζουν το υπόλοιπο/πηλίκο διαίρεσης, αντίστοιχα.

```
# Filename: selectsort.ipl
#
# Sort in ascending order an array with random
# elements, using the selection sort algorithm

# initialize
argument size args
if args == 0
        n = 1000
else
        argument 1 n
new a [n]
m = n * 9
m = m / 10
l = n / 10
while i != n
        random a[i]
        a[i] = a[i] % m
        a[i] = a[i] + l
        i = i + 1
# print
i = 0
while i != n
        j = i + 1
        m = j % 20
        if m == 0
                writeln a[i]
        else
                if j == n
                        writeln a[i]
                else
                        write a[i]
        i = j
# sort (using selectsort)
i = 0
while i != n
        min = i
        j = i + 1
        while j != n
                if a[j] < a[min]
                        min = j
                j = j + 1
        t = a[i]
        a[i] = a[min]
        a[min] = t
        i = i + 1
# print
writeln n
i = 0
while i != n
        j = i + 1
        m = j % 20
        if m == 0
                writeln a[i]
        else
                if j == n
                        writeln a[i]
                else
                        write a[i]
        i = j
free a
```

[2] Ο τύπος αυτός εξασφαλίζει ότι αν $n = 10^k$, τότε οι παραγόμενοι αριθμοί είναι με k ψηφία.

Πρόγραμμα `nqueens.ipl`: Να βρεθεί μία λύση του προβλήματος των N-βασιλισσών. Στο πρόβλημα αυτό, το ζητούμενο είναι να τοποθετηθούν σε ένα $N \times N$ πλαίσιο N βασίλισσες έτσι ώστε να μην απειλούνται μεταξύ τους, με βάση τους κανόνες στο σκάκι. Δηλαδή, να μην υπάρχουν δύο βασίλισσες στην ίδια γραμμή, στήλη ή διαγώνιο. Το N δίνεται σαν όρισμα από τη γραμμή εντολής. Αν δεν δοθεί όρισμα, να θεωρείται ότι $N = 8$ (μέγεθος τυπικής σκακιέρας).

```
# Filename: nqueens.ipl
#
# Find one solution of the N-queens problem

argument size args
if args > 0
        argument 1 n
else
        n = 8
new q[n]
# starting conditions
i = 0 - 1
ok_so_far = 1
while 0 == 0
        # go for next variable
        if ok_so_far != 0
                i = i + 1
                # check if finished
                if i == n
                        break
                # try first value
                q[i] = 0
        # try next value in same variable
        else
                q[i] = q[i] + 1
                # exhausted values
                if q[i] == n
                        # check if failed
                        if i == 0
                                break
                        # backtrack to previous variable
                        i = i - 1
                        continue
        ok_so_far = 0
        # check previous queens
        j = 0
        while j != i
                # check same column
                if q[i] == q[j]
                        continue 2
                # check same forward diagonal
                diag_i = q[i] - i
                diag_j = q[j] - j
                if diag_i == diag_j
                        continue 2
                # check same backward diagonal
                diag_i = q[i] + i
                diag_j = q[j] + j
                if diag_i == diag_j
                        continue 2
                j = j + 1
        # no collisions!
        ok_so_far = 1
# if solved print the solution
if ok_so_far != 0
        n1 = n - 1
        i = 0
        while i != n1
                x = q[i] + 1
                write x
                i = i + 1
        x = q[i] + 1
        writeln x
free q
```

Πρόγραμμα `matrmult.ipl`: Να υπολογισθεί το γινόμενο δύο διδιάστατων πινάκων A $(N \times L)$ και B $(L \times M)$ με τυχαία στοιχεία στο διάστημα $[0, 99]$. Τα N, L και M δίνονται στη γραμμή εντολής.

```
# Filename: matrmult.ipl
#
# Compute the product of an N x L matrix with an L x M matrix

argument 1 N
argument 2 L
argument 3 M
NL = N * L
LM = L * M
NM = N * M
new a[NL]
new b[LM]
new c[NM]
# random elements on first array
while i < NL
        random ele
        ele = ele % 100
        a[i] = ele
        i = i + 1
        # print element
        m = i % L
        if m != 0
                write ele
        else
                writeln ele
# random elements on second array
while j < LM
        random ele
        ele = ele % 100
        b[j] = ele
        j = j + 1
        # print element
        mod = j % M
        if mod != 0
                write ele
        else
                writeln ele
# compute and print multiplication
i = 0
while i < N
        j = 0
        while j < M
                k = 0
                # index for third (result) array
                z = i * M
                z = z + j
                # loop to compute sum
                while k < L
                        # index for first array
                        x = i * L
                        x = x + k
                        # index for second array
                        y = k * M
                        y = y + j
                        # multiply and add
                        mul = a[x] * b[y]
                        c[z] = c[z] + mul
                        k = k + 1
                j = j + 1
                # print element
                mod = j % M
                if mod != 0
                        write c[z]
                else
                        writeln c[z]
        i = i + 1
free a
free b
free c
```

Διερμηνέας της IPL

Αντικείμενο της άσκησης είναι να υλοποιήσετε ένα διερμηνέα για την IPL (έστω ότι το εκτελέσιμό του θα έχει όνομα `ipli`), δηλαδή ένα πρόγραμμα που θα παίρνει ένα IPL πρόγραμμα και θα το εκτελεί εντολή προς εντολή, διαβάζοντάς το στην πηγαία του μορφή.

Ο τρόπος κλήσης του διερμηνέα να είναι ο εξής:

```
% ./ipli [-v] <iplfilename> [<arg1>] [<arg2>] ... [<argn>]
```

Το `<iplfilename>` είναι το όνομα του πηγαίου προγράμματος IPL που θα πρέπει να εκτελέσει ο διερμηνέας. Τα `<arg1>`, `<arg2>`, ..., `<argn>` είναι τα ενδεχόμενα ορίσματα του IPL προγράμματος. Όταν δίνεται η επιλογή `-v`, οι εντολές του IPL προγράμματος εκτυπώνονται καθώς εκτελούνται, μαζί με τους αριθμούς γραμμών στις οποίες βρίσκονται στο πηγαίο πρόγραμμα.

Το αρχείο `http://www.di.uoa.gr/~ip/hwfiles/ipli/ipl` είναι ένας μεταγλωττιστής της IPL για Linux. Μπορείτε να το κατεβάσετε και να δοκιμάσετε να εκτελέσετε τα ενδεικτικά προγράμματα που δόθηκαν. Φυσικά, η ταχύτητα εκτέλεσης των προγραμμάτων από τον μεταγλωττιστή είναι πολύ μεγαλύτερη από αυτήν που θα έχει ο διερμηνέας που θα γράψετε.

Κάποια παραδείγματα εκτέλεσης του διερμηνέα δίνονται στη συνέχεια. Την ίδια ακριβώς συμπεριφορά έχει και ο μεταγλωττιστής (εκτός από την έξοδο όταν δίνεται η επιλογή `-v`).

```
% ./ipli primes.ipl
20
2
3
5
7
11
13
17
19
%
% ./ipli -v primes.ipl
   5:   read maxnumb
10
   6:   number = 1
   7:   while number <= maxnumb
   8:           if number == 1
   9:                   is_prime = 0
  10:           else
  12:           divisor = 2
  13:           if number < 4
  14:                   stop = 1
  15:           else
  17:           while stop == 0
  27:           if is_prime != 0
  29:           number = number + 1
   7:   while number <= maxnumb
   8:           if number == 1
  10:           else
  11:                   is_prime = 1
```

```
  12:           divisor = 2
  13:           if number < 4
  14:                   stop = 1
  15:           else
  17:           while stop == 0
  27:           if is_prime != 0
  28:                   writeln number
2
  29:           number = number + 1
   7:   while number <= maxnumb
   8:           if number == 1
  10:           else
  11:                   is_prime = 1
  12:           divisor = 2
  13:           if number < 4
  14:                   stop = 1
  15:           else
  17:           while stop == 0
  27:           if is_prime != 0
  28:                   writeln number
3
  29:           number = number + 1
   7:   while number <= maxnumb
   8:           if number == 1
  10:           else
  11:                   is_prime = 1
.................................................
  29:           number = number + 1
   7:   while number <= maxnumb
%
% ./ipli happypr.ipl
677323349
515814941
242697757
534585479
242183587
861705041
913789361
221397541
19535209
174886339
103743349
160804873
787834871
21679897
240314629
7580161
391503883
137573453
```

```
90244981
523540711
112983469
867079621
1100051
270018209
.................................
%
% ./ipli selectsort.ipl 100
13 95 21 48 93 37 83 85 38 41 80 28 88 12 70 66 40 41 32 20
85 41 43 39 45 63 47 85 17 77 86 72 34 97 72 27 87 17 12 25
48 82 43 89 84 65 17 76 96 39 86 43 70 82 73 68 45 20 53 14
49 39 39 73 89 11 52 38 19 55 53 19 89 48 98 36 13 15 12 10
96 51 43 29 33 68 87 30 78 40 34 79 31 63 14 72 65 57 10 36
100
10 10 11 12 12 12 13 13 14 14 15 17 17 17 19 19 20 20 21 25
27 28 29 30 31 32 33 34 34 36 36 37 38 38 39 39 39 39 40 40
41 41 41 43 43 43 43 45 45 47 48 48 48 49 51 52 53 53 55 57
63 63 65 65 66 68 68 70 70 72 72 72 73 73 76 77 78 79 80 82
82 83 84 85 85 85 86 86 87 87 88 89 89 89 93 95 96 96 97 98
%
% ./ipli nqueens.ipl 15
1 3 5 2 10 12 14 4 13 9 6 15 7 11 8
%
% ./ipli matrmult.ipl 6 4 7
47 42 58 28
26 85 87 68
46 55 79 33
97 1 21 41
69 28 63 82
61 84 42 1
65 66 17 71 44 82 92
91 25 2 71 3 88 11
72 86 66 3 72 15 4
93 56 25 74 20 7 35
13657 10708 5411 8565 6930 8616 5998
22013 15131 8054 13174 9023 11393 6055
16752 13053 6931 9850 8537 10028 6308
11721 10529 4062 10055 6603 8644 10454
19195 15264 7437 13144 9296 9641 9778
14726 9794 4002 10495 5980 13031 6739
%
```

Παραδοτέο

Το πρόγραμμα που θα γράψετε θα πρέπει να είναι δομημένο σε ένα σύνολο από **τουλάχιστον τρία πηγαία αρχεία** C (με κατάληξη `.c`) και **τουλάχιστον δύο αρχεία επικεφαλίδας** (με κατάληξη `.h`). Για να παραδώσετε τη δουλειά σας, θα πρέπει να κάνετε τα εξής:

- Τοποθετήστε όλα τα αρχεία (πηγαία και αρχεία επικεφαλίδας) μέσα σ΄ ένα κατάλογο που θα δημιουργήσετε, έστω με όνομα `ipli`, σε κάποιο σύστημα Unix. Επίσης, τοποθετήστε στον κατάλογο αυτό και ένα αρχείο με όνομα `README`, στο οποίο να δίνετε οδηγίες για τη μεταγλώττιση των αρχείων και την κατασκευή του τελικού εκτελέσιμου, καθώς και ό,τι άλλο κρίνετε σκόπιμο να επισημάνετε. Αν για τον έλεγχο της ορθής λειτουργίας του διερμηνέα που υλοποιήσατε γράψατε και επιπλέον IPL προγράμματα, τοποθετήστε τα και αυτά μέσα στον κατάλογο προς παράδοση. Προαιρετικά, μπορείτε να παραδώσετε και ένα αρχείο `Makefile` που να αναλαμβάνει όλη τη διαδικασία της κατασκευής του τελικού εκτελέσιμου μέσω της εντολής "`make`" (δώστε "`man make`" για περισσότερες λεπτομέρειες).

- Όντας στον κατάλογο που περιέχει τον κατάλογο `ipli`, δημιουργήστε ένα επιπεδοποιημένο `tar` αρχείο (έστω με όνομα `ipli.tar`) που περιέχει τον κατάλογο `ipli` και όλα του τα περιεχόμενα. Αυτό γίνεται με την εντολή "`tar cvf ipli.tar ipli`".[3]

- Συμπιέστε το αρχείο `ipli.tar`, ώστε να δημιουργηθεί το αρχείο `ipli.tar.gz`. Αυτό γίνεται με την εντολή "`gzip ipli.tar`".[4]

- Το αρχείο `ipli.tar.gz` είναι που θα πρέπει να υποβάλετε, με διαδικασία που θα ανακοινωθεί σύντομα.

Ομαδική άσκηση

Η άσκηση αυτή μπορεί να παραδοθεί και από **ομάδες των δύο ατόμων**. Στην περίπτωση αυτή, θα παραδοθεί μόνο από το ένα μέλος της ομάδας, αλλά μέσα στο αρχείο `README` θα αναφέρονται σαφώς τα στοιχεία των δύο μελών. Ο στόχος της διαδικασίας αυτής είναι να ενισχυθεί η ιδέα της **ισότιμης** συνεργασίας σε μία ομάδα για την επίτευξη ενός στόχου. Αν τα μέλη της ομάδας έχουν υλοποιήσει διαφορετικά τμήματα της άσκησης, θα πρέπει στο αρχείο `README` να αναφέρεται ρητά τι έχει υλοποιήσει κάθε μέλος, έτσι ώστε στην προφορική εξέταση που θα ακολουθήσει, να μην υπάρχει η απαίτηση να έχει κάποιο μέλος της ομάδας πλήρη γνώση του πώς έχουν υλοποιηθεί τα τμήματα στα οποία εκείνο δεν έχει εμπλακεί.

[3] Αν θέλετε να ανακτήσετε την δενδρική δομή που έχει φυλαχθεί σ΄ ένα επιπεδοποιημένο `tar` αρχείο `file.tar`, αυτό μπορεί να γίνει με την εντολή "`tar xvf file.tar`".

[4] Αν θέλετε να αποσυμπιέσετε ένα αρχείο `file.gz` που έχει συμπιεσθεί με την εντολή `gzip`, αυτό μπορεί να γίνει με την εντολή "`gzip -d file.gz`".