ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
Εθνικόν και Καποδιστριακόν Πανεπιστήμιον Αθηνών
ΙΔΡΥΘΕΝ ΤΟ 1837

Εισαγωγή στον Προγραμματισμό

Εργασία #2

Δεκέμβριος 2023

Με τις πρώτες μας εργασίες γράψαμε προγράμματα που διαχειρίζονται αριθμούς (ακεραίους και κινητής υποδιαστολής). Σε αυτήν την εργασία θα συνεχίσουμε να χτίζουμε την εμπειρία μας με άλλους τύπους, προγράμματα και μεθόδους εισόδου. Συγκεκριμένα, στοχεύουμε σε εμπειρία με τα ακόλουθα:

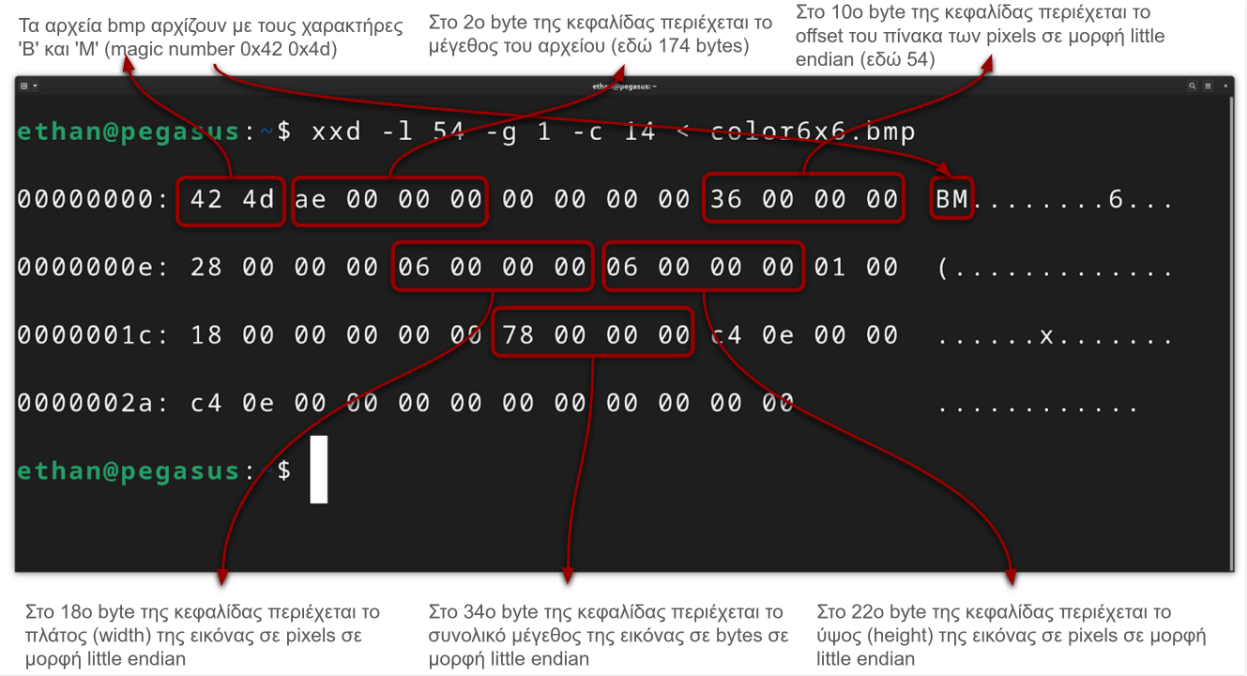
1. Χρήση πινάκων, χαρακτήρων και συμβολοσειρών
2. Χρήση δεικτών και δυναμικής μνήμης
3. Διάβασμα δεδομένων εισόδου από stdin
4. Διάβασμα δεδομένων εισόδου από αρχείο
5. Έκθεση σε προγραμματιστικές τεχνικές

Υποβολή Εργασίας. Όλες οι υποβολές των εργασιών θα γίνουν μέσω GitHub και συγκεκριμένα στο github.com/progintro [1]. Προκειμένου να ξεκινήσεις, μπορείς να δεχτείς την άσκηση με αυτήν την: πρόσκληση [2].

1 FauxtoShop (50 Μονάδες)

Πολύ συχνά βγάζουμε φωτογραφίες με το κινητό ή την φωτογραφική μηχανή (για πιο ρετρό αποτέλεσμα) αλλά δεν έχουν τον σωστό προσανατολισμό και θέλουμε να τις περιστρέψουμε. Κάποια προγράμματα επεξεργασίας εικόνας μας επιτρέπουν να κάνουμε τέτοιες περιστροφές αλλά πολλές φορές κοστίζουν ή έχουν περιορισμούς στο μέγεθος της εικόνας [7]. Δεν θέλουμε να έχουμε τέτοιους περιορισμούς!

Το ζητούμενο σε αυτήν την άσκηση είναι να υλοποιήσουμε το δικό μας fauxtoshop που μας επιτρέπει να περιστρέψουμε εικόνες τύπου BMP (BitMaP) [10] δεξιόστροφα κατά 90^o (με την φορά του ρολογιού). Προκειμένου να το καταφέρουμε αυτό, θα χρειαστεί να μάθουμε λίγο παραπάνω για την μορφή των αρχείων BMP. Σε αυτήν την



Σχήμα 1: Στο παραπάνω τερματικό τρέχουμε την εντολή xxd και τυπώνουμε τα 54 πρώτα bytes της κεφαλίδας του αρχείου. Στην συνέχεια δείχνουμε κάποια από τα καίρια στοιχεία της κεφαλίδας (απαραίτητα για την άσκηση) και πως αυτά ερμηνεύονται από τυπικά προγράμματα επεξεργασίας εικόνας.

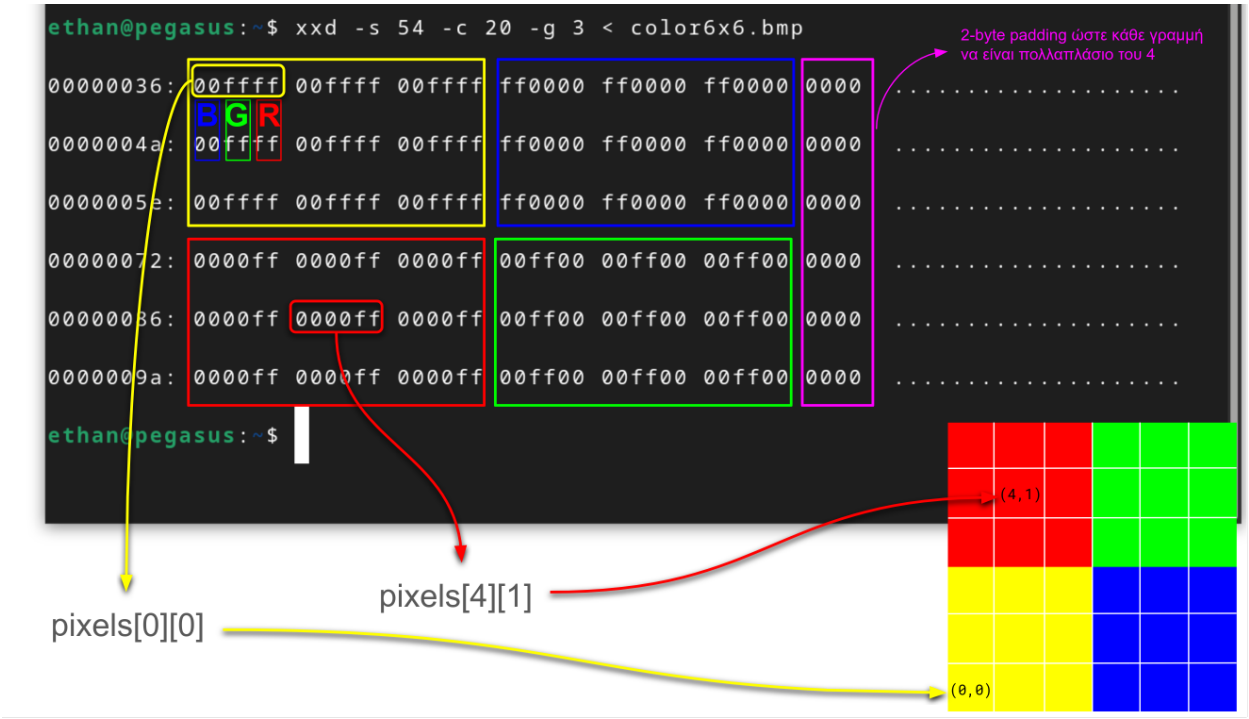
άσκηση θα ασχοληθούμε με μια κατηγορία αρχείων BMP τα οποία έχουν την μορφή κεφαλίδων (headers) ακολουθούμενη από κάποια κάποια bytes και έναν δισδιάστατο πίνακα εικονοστοιχείων (pixels):

$$\text{Headers (H bytes)} \mid \text{Other Data (M bytes)} \mid \text{Pixel 2-D array (N bytes)}$$

Τα Σχήματα 1 και 2 παρουσιάζουν την ανάλυση των περιεχομένων ενός ενδεικτικού αρχείου color6x6.bmp. Για τις ανάγκες της άσκησης, μπορείτε να υποθέσετε ότι $H \geq 54$ και ότι τα δεδομένα ανάμεσα στις κεφαλίδες Other Data δεν χρειάζονται αλλαγή αλλά πρέπει να αντιγραφούν όπως είναι. Μπορείτε να μάθετε περισσότερα για την δομή των αρχείων BMP στο διαδίκτυο [10] και στις τεχνικές προδιαγραφές που προσδιορίζουμε τα αρχεία που θα ελέγξουμε σε αυτήν την άσκηση.

Τεχνικές Προδιαγραφές

- Repository Name: progintro/hw2-<YourUsername>
- C Filepath: fauxtoshop/src/fauxtoshop.c
- Ένα αρχείο εισόδου/εξόδου BMP είναι δεκτό μόνο εάν:



Σχήμα 2: Με την εντολή xxd παραλείπουμε τα 54 πρώτα στοιχεία της κεφαλίδας του αρχείου και τυπώνουμε μόνο τα bytes που αντιστοιχούν στα pixel της εικόνας. Στο σχέδιο παραπάνω βλέπουμε πως τα bytes ερμηνεύονται ως κώδικας RGB και πως διατάσσονται σε ένα δισδιάστατο πλέγμα για να δούμε την εικόνα με κάποιον viewer. Δίνουμε **προσοχή στο padding** που μπορεί να έχει το κάθε αρχείο bmp το οποίο δεν έχει απεικόνιση για τους χρήστες. Το αρχείο color6x6.bmp εικονίζεται κάτω δεξιά σε μεγεθυμένη μορφή προκειμένου να φαίνονται τα pixel.

  - Ξεκινάει την μαγική κεφαλίδα με τα bytes: 'B' 'M'.
  - Χρησιμοποιεί 24-bit για την αναπαράσταση χρώματος (μονοχρωματικά BMP δεν είναι δεκτά).
  - Χρησιμοποιεί τουλάχιστον 54 (14 + 40) bytes για την αναπαράσταση των κεφαλίδων ακολουθώντας το Windows 3.x format (Bitmap file header + DIB header). Για παράδειγμα, αρχεία που είναι συμβατά με την μορφή κεφαλίδας που δείχνουμε στο Σχήμα 1 είναι δεκτά.
  - Έχει σωστό μέγεθος αρχείου (τα μεγέθη στις κεφαλίδες συμφωνούν με τα περιεχόμενα του αρχείου).
  - Έχει σωστό μήκος, πλάτος και padding.

- Το πρόγραμμά θα πρέπει να παίρνει είσοδο από το standard input. Για οποιαδήποτε είσοδο δεν είναι μέσα στις προδιαγραφές το πρόγραμμα πρέπει να επιστρέφει με κωδικό εξόδου (exit code) 1.

- Ένα αρχείο `image.bmp` μπορεί να περαστεί στο πρόγραμμά μας με ανακατεύθυνση (redirection) στο stdin και αντίστοιχα η έξοδος μπορεί να γραφτεί σε ένα αρχείο rotated.bmp και πάλι με ανακατεύθυνση του stdout:

```
$ ./fauxtoshop < image.bmp > rotated.bmp
```

- Το αρχείο C που θα υποβληθεί πρέπει να μεταγλωττίζεται χωρίς ειδοποιήσεις για λάθη και με κωδικό επιστροφής (exit code) που να είναι 0. Συγκεκριμένα, το αρχείο σας **πρέπει** να μπορεί να μεταγλωττιστεί επιτυχώς με την ακόλουθη εντολή σε ένα από τα μηχανήματα του εργαστηρίου (linuxXY.di.uoa.gr):

```
gcc -O3 -Wall -Wextra -Werror -pedantic -o fauxtoshop fauxtoshop.c
```

- README Filepath: fauxtoshop/README.md

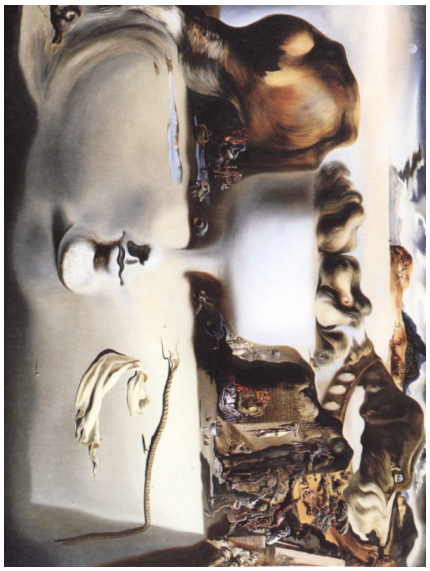
- **Ένα αρχείο που να περιέχει στοιχεία εισόδου και ένα εξόδου διαφορετικά από αυτά της άσκησης. Συγκεκριμένα προτείνουμε να βάλετε έναν συνδυασμό που θεωρείται ότι είναι δύσκολος να γίνει σωστός**.
  - input Filepath: fauxtoshop/test/input.bmp
  - output Filepath: fauxtoshop/test/output.bmp

- Πρέπει να ολοκληρώνει την εκτέλεση μέσα σε: 10 δευτερόλεπτα.

- **Δεν επιτρέπεται** η χρήση δομών/εγγραφών (`struct`).

Παρακάτω παραθέτουμε την αλληλεπίδραση με μια ενδεικτική λύση:

Αρχική εικόνα πριν την περιστροφή.



Εικόνα μετά την περιστροφή.

Σχήμα 3: Χρήση του προγράμματος fauxtoshop σε ενδεικτική εικόνα.
`./fauxtoshop < Apparition_of_Face_and_Fruit.bmp > app90.bmp`

```
$ gcc -O3 -Wall -Wextra -Werror -pedantic -o fauxtoshop fauxtoshop.c
$ ./fauxtoshop < ./fauxtoshop
Error: not a BMP file
$ echo $?
1
$ ./fauxtoshop < color6x6.bmp > color6x6.90.bmp
$ ./fauxtoshop < color6x6.90.bmp > color6x6.180.bmp
$ ./fauxtoshop < color6x6.180.bmp > color6x6.270.bmp
$ ./fauxtoshop < color6x6.270.bmp > color6x6.360.bmp
$ md5sum color6x6*.bmp
755c07991c51c0b3af855ba7f668c7d0  color6x6.180.bmp
5f19168fa9511a6520ec5dba2db4b478  color6x6.270.bmp
33855d8642cf110f74464f8ff78449aa  color6x6.360.bmp
6d62a0c4264dbfa962b372b6e5c8b9be  color6x6.90.bmp
33855d8642cf110f74464f8ff78449aa  color6x6.bmp
$ wc -c color6x6*.bmp
174 color6x6.180.bmp
174 color6x6.270.bmp
174 color6x6.360.bmp
174 color6x6.90.bmp
174 color6x6.bmp
$ ./fauxtoshop < Apparition_of_Face_and_Fruit.bmp > app90.bmp
$ md5sum Apparition_of_Face_and_Fruit.bmp app90.bmp
841d3634ba8c6af641d86b63ac1b6dfc  Apparition_of_Face_and_Fruit.bmp
```

```
96e8c33582f8a922a6665d4891e62a77  app90.bmp
$ wc -c Apparition_of_Face_and_Fruit.bmp app90.bmp
1440054 Apparition_of_Face_and_Fruit.bmp
1440054 app90.bmp
2880108 total
$ ./fauxtoshop < school_of_athens.bmp > school_of_athens90.bmp
$ md5sum school_of_athens*.bmp
282d01bbd23044a7e6ca91f0dda2238f  school_of_athens90.bmp
112ce7f3a60265992dc939949232257d  school_of_athens.bmp
$ wc -c school_of_athens*.bmp
15252534 school_of_athens.bmp
15257654 school_of_athens90.bmp
30510188 total
$ file school_of_athens.bmp school_of_athens90.bmp
school_of_athens.bmp:   PC bitmap, Windows 3.x format, 2560 x 1986 x 24,
                        image size 15252480, cbSize 15252534, bits offset 54
school_of_athens90.bmp: PC bitmap, Windows 3.x format, 1986 x 2560 x 24,
                        image size 15257600, cbSize 15257654, bits offset 54
$ ./fauxtoshop < school_of_athens.bmp |./fauxtoshop | ./fauxtoshop |
           ./fauxtoshop > school_of_athens360.bmp
$ md5sum school_of_athens*.bmp
112ce7f3a60265992dc939949232257d  school_of_athens360.bmp
282d01bbd23044a7e6ca91f0dda2238f  school_of_athens90.bmp
112ce7f3a60265992dc939949232257d  school_of_athens.bmp
```

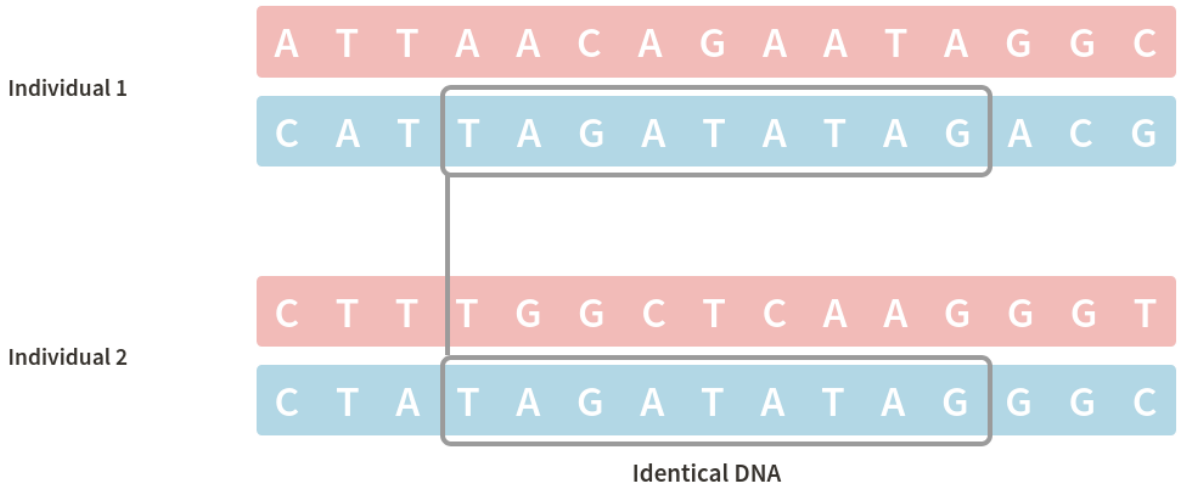
Το Σχήμα 3 δείχνει μία από τις εικόνες παραδείγματα μετά την περιστροφή της. Τα παραπάνω bmp αρχεία είναι στο: https://github.com/progintro/data/tree/main/bmp άλλα μπορείτε να δοκιμάσετε και άλλα παραδείγματα εικόνων που βρίσκετε online. Εκτός από scp για να τα αντιγράψετε στο Linux lab, μπορείτε να τα κατεβάσετε και μέσω κονσόλας χρησιμοποιώντας curl, για παράδειγμα:

```
$ curl -O https://raw.githubusercontent.com/progintro/data/main/bmp/color6x6.bmp
```

Ως συνήθως, στο αρχείο README.md πρέπει να προσθέσετε οποιεσδήποτε παρατηρήσεις σας κατά την διεκπεραίωση της άσκησης. Ο κώδικας απαιτείται να είναι καλά τεκμηριωμένος με σχόλια καθώς αυτό θα είναι μέρος της βαθμολόγησης. **Δεν αφήνουμε το γράψιμο των README/σχολίων για τελευταία στιγμή!**

2 DNA Matching (50 Μονάδες)

Ένα εργαστήριο βιοπληροφορικής επικοινώνησε πρόσφατα με την σχολή μας και μας ενημέρωσε ότι ψάχνει επειγόντως λύσεις για να βρίσκει *αποδοτικά* κοινές αλληλουχίες DNA ανάμεσα σε διαφορετικά άτομα ή ακόμα και οργανισμούς, γνωστό



Σχήμα 4: Εύρεση κοινής αλυσίδας DNA ανάμεσα σε δύο άτομα.

και ως DNA matching [5]. Τους είπαμε ότι θα το κοιτάξουμε άμεσα και θα τους προσφέρουμε λύσεις καθώς είμαστε εξπέρ στον προγραμματισμό. Το πρόβλημα; Δεν ξέρουμε τίποτα για DNA! Ας είναι καλά η wikipedia.

Το DNA [4] περιέχει τις γενετικές οδηγίες - τον κώδικα! - για την ανάπτυξη και την λειτουργία όλων των γνωστών οργανισμών. Συνήθως το αναπαριστούμε με την μορφή μιας αλυσίδας (συμβολοσειράς) από τέσσερα διαφορετικά στοιχεία που λέγονται *βάσεις*: A (Αδενίνη), G (Γουανίνη), T (Θυμίνη) και C (Κυτοσίνη) τα οποία μπορούν να συνδυαστούν με οποιαδήποτε σειρά. Σε αντίθεση με τον δυαδικό κώδικα που κάθε στοιχείο ενός δυαδικού αριθμού είναι 0 ή 1, κάθε στοιχείο της αλυσίδας DNA είναι A ή G ή T ή C. Οργανισμοί του ίδιου είδους συνήθως έχουν μοναδική αλυσίδα DNA αλλά με πολλές ομοιότητες μεταξύ τους. Πολλές ομοιότητες συνήθως εμφανίζονται ακόμα και σε οργανισμούς διαφορετικού είδους! Προκειμένου να κάνουμε DNA matching θέλουμε να δούμε αν υπάρχουν κοινά τμήματα σε αυτές τις αλυσίδες DNA και να δούμε πόσο μεγάλες είναι / σε ποια γονίδια αντιστοιχούν αυτά τα κοινά τμήματα. Το σχήμα 4 δείχνει ένα παράδειγμα δύο αλυσίδων με κοινό τμήμα.

Για το ζητούμενο αυτής της άσκησης, καλείστε να γράψετε ένα πρόγραμμα που διαβάζει δύο αλυσίδες DNA, βρίσκει την μέγιστη κοινή αλυσίδα ανάμεσα στις δύο και την τυπώνει.

Τεχνικές Προδιαγραφές

- Repository Name: progintro/hw2-<YourUsername>
- C Filepath: dna/src/dna.c
- Το πρόγραμμά σας θα πρέπει να παίρνει δύο ορίσματα από την γραμμή εντο-

λών. Και τα δύο θα είναι ονόματα αρχείων που περιέχουν τις εισόδους για το πρόγραμμα, δηλαδή τις δύο αλυσίδες DNA - στην μορφή `./dna filename1 filename2`. Για το περιεχόμενο των αρχείων δείτε παρακάτω. Αν δεν περαστούν δύο ορίσματα στην γραμμή εντολών ή οποιοδήποτε από τα αρχεία δεν μπορεί να ανοιχτεί το πρόγραμμα πρέπει να επιστρέφει με κωδικό εξόδου (exit code) 1.

- Η βασική είσοδος για το πρόγραμμα θα είναι μέσω των περιεχομένων των αρχείων. Τα αρχεία θα περιέχουν τις αλυσίδες DNA οι οποίες αποτελούνται από τους κεφαλαίους χαρακτήρες A, G, T και C. Όλοι οι χαρακτήρες θα είναι σε μορφή ASCII, και το αρχείο θα διαβάζεται και ως κείμενο. Οποιοσδήποτε χαρακτήρας εκτός από τους A, G, T ,C πρέπει να αγνοηθεί καθώς θεωρείται σφάλμα που δημιουργήθηκε κατά το sequencing [6] της αλυσίδας.

- Η έξοδος του προγράμματος θα πρέπει να είναι η μέγιστη κοινή αλυσίδα των δύο αλυσίδων που δώσαμε ως είσοδο, ακολουθούμενη από μια καινούρια γραμμή. Όλα τα στοιχεία της αλυσίδας εξόδου θα πρέπει να είναι βάσεις DNA - δεν επιτρέπονται άλλοι χαρακτήρες. Εάν υπάρχουν πάνω από μία μέγιστη κοινή αλυσίδα, αρκεί να επιστραφεί μία από αυτές.

- Το αρχείο C που θα υποβληθεί πρέπει να μεταγλωττίζεται χωρίς ειδοποιήσεις για λάθη και με κωδικό επιστροφής (exit code) που να είναι 0. Συγκεκριμένα, το αρχείο σας **πρέπει** να μπορεί να μεταγλωττιστεί επιτυχώς με την ακόλουθη εντολή σε ένα από τα μηχανήματα του εργαστηρίου (linuxXY.di.uoa.gr):

  ```
  gcc -Ofast -m32 -Wall -Wextra -Werror -pedantic -o dna dna.c -lm
  ```

- README Filepath: dna/README.md

- **Ένα αρχείο που να περιέχει στοιχεία εισόδου και ένα εξόδου διαφορετικά από αυτά της άσκησης. Συγκεκριμένα προτείνουμε να βάλετε έναν συνδυασμό που θεωρείται ότι είναι δύσκολος να γίνει σωστός**.
  - input1 Filepath: dna/test/input1.dna
  - input2 Filepath: dna/test/input2.dna
  - output Filepath: dna/test/output.dna

- Πρέπει να ολοκληρώνει την εκτέλεση μέσα σε: 90 δευτερόλεπτα.

- Το μέγιστο μέγεθος επιτρεπτής αλυσίδας: 100.000 βάσεις.

Παρακάτω παραθέτουμε την αλληλεπίδραση με μια ενδεικτική λύση με μερικά δείγματα DNA που κατεβάσαμε (η ονοματολογία που δίνουν οι επιστήμονες σε αυτούς τους οργανισμούς είναι μερικές φορές αρκετά ευφάνταστη):

```
$ gcc -Ofast -Wall -Wextra -Werror -pedantic -o dna dna.c -lm
$ wc -c *.dna
 62487 alien.dna
  8193 carsonella-ruddii.dna
 47811 claviceps-purpurea.dna
 21992 escherichia-coli.dna
    16 sample1.dna
    16 sample2.dna
 99492 theobroma-cacao.dna
240007 total
$ $./dna
Error: arguments missing. Usage: ./dna dnafile1 dnafile2
$ echo $?
1
$ ./dna3 foo.bar bar.zonk
Error: cannot open file foo.bar
$ echo $?
1
$ cat sample1.dna
CATTAGATATAGACG
$ cat sample2.dna
CTATAGATATAGGGC
$ ./dna sample1.dna sample2.dna
TAGATATAG
$ echo -e "CTATAGAT\nHello WORLDATAGGG" > split.dna
$ ./dna split.dna split.dna
CTATAGATATAGGG
$ ./dna carsonella-ruddii.dna sample1.dna
ATATAGACG
$ ./dna escherichia-coli.dna carsonella-ruddii.dna
AATTAAAATTTTATT
$ ./dna claviceps-purpurea.dna carsonella-ruddii.dna
GTTTTTTTTTTCT
$ time ./dna theobroma-cacao.dna escherichia-coli.dna
GGTTTGCTTTTATG

real 0m4.550s
user 0m4.549s
sys 0m0.001s
$ time ./dna theobroma-cacao.dna alien.dna
AAAAAAAAAAAAAAAAACC

real 0m2.828s
user 0m2.827s
sys 0m0.001s
```

```
$ time ./dna theobroma-cacao.dna theobroma-cacao.dna > shared.dna

real 0m21.817s
user 0m21.815s
sys 0m0.001s
$ wc -c shared.dna
98165 shared.dna
$ md5sum shared.dna
5ca8c9e6fd76d46221d3beadd610b165  shared.dna
$ time ./dna alien.dna alien.dna > shared.dna

real 0m1.917s
user 0m1.915s
sys 0m0.001s
$ wc -c shared.dna
62488 shared.dna
$ md5sum shared.dna
d4544ab6971c6a54bb375159adc5852b  shared.dna
```

Τα παραπάνω dna αρχεία είναι στο: https://github.com/progintro/data/tree/main/dna άλλα μπορείτε να δοκιμάσετε και άλλα παραδείγματα δικά σας. Στο αρχείο README.md πρέπει να προσθέσετε οποιεσδήποτε παρατηρήσεις σας κατά την διεκπεραίωση της άσκησης. Ο κώδικας απαιτείται να είναι καλά τεκμηριωμένος με σχόλια καθώς (σωστά το μαντέψατε) αυτό θα είναι μέρος της βαθμολόγησης.

3 Συνεργασία (50 Μονάδες)

Πολλά επιτραπέζια παιχνίδια βασίζονται στην δυνατότητα να συνεργαστείς (ή όχι) με κάποιον άλλο και ανάλογα με τις επιλογές σου μπορεί να τα πας καλά (ή όχι) στο παιχνίδι. Πολλές φορές, αυτός ο "κοινωνικός" παράγοντας μπορεί να αλλάξει εντελώς το παιχνίδι ακόμα και όταν δεν περιγράφεται στους κανόνες του παιχνιδιού. Για παράδειγμα, έστω ότι παίζουμε Uno [11][1] και πρέπει να ρίξω μια κάρτα στον διπλανό μου - μπορώ να ρίξω μια κάρτα που του στερεί την σειρά του ή μπορώ να ρίξω μια κάρτα που τον βοηθάει. Η επιλογή μου πιθανόν να καθορίσει πως θα συμπεριφερθεί ο διπλανός μου (ή και οι άλλοι παίκτες) στην συνέχεια του παιχνιδιού. Για παράδειγμα, μπορεί όλοι να στραφούν εναντίον μου ή μπορεί να με βοηθήσουν και εμένα να ξεφορτωθώ τις κάρτες μου. Ποια είναι η καλύτερη στρατηγική;

Αντίστοιχες ερωτήσεις και δυναμικές εμφανίζονται σε πολλές εκφάνσεις του κόσμου μας, από τα οικονομικά μέχρι την ψυχολογία, και υπάρχουν πεδία της πληροφορικής που αναζητούν απαντήσεις όπως η θεωρία παιγνίων (game theory) [8, 3].

[1]Αν δεν γνωρίζετε το παιχνίδι δεν έχει πολλή σημασία, απλά πρέπει να ξέρετε ότι έχει πολλούς γύρους και σε κάθε γύρο ο κάθε παίκτης ρίχνει μια κάρτα που πιθανώς βοηθάει τον διπλανό του ή τον τιμωρεί.

Ένας τρόπος να μοντελοποιήσουμε τέτοια παιχνίδια με πολλούς γύρους όπου οι παίκτες συνεργάζονται ή όχι είναι το Prisoner's Dilemma [9]. Σε κάθε γύρο αυτού του παιχνιδιού, δύο παίκτες Α και Β αποφασίζουν αν θα συνεργαστούν (cooperate) ή όχι (defect). Αν και οι δύο συνεργαστούν τότε αποκομίζουν κάποιους πόντους (έστω Reward (R) = 3). Αν και οι δύο δεν συνεργαστούν τότε παίρνουν λιγότερους πόντους (έστω Payoff (P) = 1). Αν ο ένας (έστω ο Α) αποφασίσει να συνεργαστεί και ο Β τον "προδώσει" χωρίς να συνεργαστεί τότε ο Β θα αποκομίσει περισσότερους πόντους (έστω Temptation (T) = 5) ενώ ο Α δεν θα πάρει καθόλου πόντους (έστω Sucker (S) = 0). Ένας τρόπος να συνοψίσουμε το παραπάνω παιχνίδι πόντων είναι με έναν πίνακα (payoff matrix):

A / B	B συνεργάζεται	B δεν συνεργάζεται
A συνεργάζεται	A: 3, B: 3	A: 0, B: 5
A δεν συνεργάζεται	A: 5, B: 0	A: 1, B: 1

Όμως τα περισσότερα παιχνίδια δεν τελειώνουν στον πρώτο γύρο! Για παράδειγμα, αν στον πρώτο γύρο είδε ο Α ότι ο Β δεν συνεργάστηκε, ο Α μπορεί να αποφασίσει ότι δεν θα συνεργαστεί στον δεύτερο γύρο και θα περιμένει να δει πως θα "συμπεριφερθεί" ο Β στον δεύτερο γύρο. Αντίστοιχες αποφάσεις μπορεί να έχει στο μυαλό του και ο Β.

Για το ζητούμενο αυτής της άσκησης, ο στόχος είναι να γράψετε ένα πρόγραμμα που θα αποφασίζει πως να συμπεριφερθεί αποδοτικά σε τέτοια επαναληπτικά παιχνίδια αποφάσεων.

Τεχνικές Προδιαγραφές

- Repository Name: progintro/hw2-<YourUsername>
- C Filepath: coop/src/coop.c
- Το πρόγραμμα θα πρέπει να δέχεται από την πρότυπη είσοδο (stdin) 2 χαρακτήρες: 'C' (Cooperate), 'D' (Defect / Don't cooperate) - όλοι οι άλλοι χαρακτήρες αγνοούνται. Το πρόγραμμα θα πρέπει να τερματίζει με exit code 0 όταν λάβει EOF.
- Όταν ξεκινήσει το πρόγραμμά μας πρέπει να τυπώσει την εντολή που επιθυμεί **πριν** επιχειρήσει να διαβάσει την εντολή του άλλου παίκτη. Η εντολή του άλλου παίκτη παρέχεται μόνο αφού το πρόγραμμά σας τυπώσει την δική του εντολή.
- Το πρόγραμμα θα πρέπει να τυπώνει εντολές ως χαρακτήρες σε δύο μορφές: 'C' (Cooperate) και 'D' (Defect / Don't Cooperate). Μετά από κάθε εντολή το πρόγραμμά σας **πρέπει** να τυπώνει καινούρια γραμμή ('\n').
- Για την είσοδο και έξοδο του προγράμματος μπορούν να χρησιμοποιηθούν **μόνο** οι συναρτήσεις `getchar` και `putchar`.

- Το πρόγραμμά σας **πρέπει** να τυπώνει σε κάθε γύρο την απόφασή του ανεξαρτήτως του buffering που μπορεί να υπάρχει - συνίσταται η χρήση της συνάρτησης `fflush` μετά από κάθε κλήση της `putchar`.
- Το αρχείο C που θα υποβληθεί πρέπει να μεταγλωττίζεται χωρίς ειδοποιήσεις για λάθη και με κωδικό επιστροφής (exit code) που να είναι 0. Συγκεκριμένα, το αρχείο σας **πρέπει** να μπορεί να μεταγλωττιστεί επιτυχώς με την ακόλουθη εντολή σε ένα από τα μηχανήματα του εργαστηρίου (linuxXY.di.uoa.gr):

  ```
  gcc -Os -Wall -Wextra -Werror -pedantic -o coop coop.c -lm
  ```

- Το πρόγραμμά μας πρέπει να παίρνει τουλάχιστον 1 πόντο σε κάθε 1000 γύρους με έναν εκδικητικό παίκτη.
- Το πρόγραμμά μας πρέπει να παίρνει τουλάχιστον 3000 πόντους σε κάθε 1000 γύρους με έναν συνεργατικό παίκτη.
- Το πρόγραμμά μας πρέπει να παίζει τουλάχιστον μία φορά D και μία φορά C σε κάθε 1000 γύρους παιχνιδιού.
- README Filepath: coop/README.md
- Μέγιστος αριθμός γύρων: 10^6.
- Πρέπει να ολοκληρώνει την εκτέλεση 1000 εντολών μέσα σε: 1 δευτερόλεπτο.

Ας προσπαθήσουμε να παίξουμε με μια ενδεικτική λύση (όχι η καλύτερη):

```
$ gcc -Os -Wall -Wextra -Werror -pedantic -o coop coop.c -lm
$ ./coop
C
```

Παρατηρούμε ότι το πρόγραμμα έπαιξε "C" (Cooperate). Σε απάντηση μπορούμε να απαντήσουμε D και μετά Enter:

```
$ gcc -Os -Wall -Wextra -Werror -pedantic -o coop coop.c -lm
$ ./coop
C
D
D
```

Παρατηρούμε ότι αφού γράψαμε "D" (Defect) το πρόγραμμα μας απάντησε με "D" για τον επόμενο γύρο. Έστω ότι απαντάμε D σε αυτόν τον γύρο και εμείς. Σύνολο σε αυτούς τους δύο γύρους επομένως εμείς (που γράφουμε από το stdin) συλλέξαμε 5 + 1 = 6 πόντους ενώ η ενδεικτική μας λύση (coop) μάζεψε 0 + 1 = 1 πόντους.

Έστω ότι θέλουμε να δούμε πως συμπεριφέρεται μια ενδεικτική (όχι η καλύτερη) λύση όταν παίζει με έναν "εκδικητικό" παίκτη που πάντα δεν συνεργάζεται:

```
$ echo > input; for i in `seq 1 1000`; do echo D >> input; done
$ ./coop < input | head -n -1 > output
$ grep -c D output
42
$ grep -c C output
958
```

Επομένως παρατηρούμε ότι το πρόγραμμά μας συνεργάστηκε 958 φορές και 42 δεν συνεργάστηκε, επομένως συγκέντρωσε $958 \cdot 0 + 42 = 42$ πόντους.

Αντίστοιχα μπορούμε να ελέγξουμε πως συμπεριφέρεται με έναν "συνεργατικό" παίκτη:

```
$ echo > input; for i in `seq 1 1000`; do echo C >> input; done
$ ./coop < input | head -n -1 > output
$ grep -c D output
3
$ grep -c C output
997
```

Επομένως παρατηρούμε ότι το πρόγραμμά μας συνεργάστηκε 997 φορές και 3 δεν συνεργάστηκε, επομένως συγκέντρωσε $997 \cdot 3 + 3 \cdot 5 = 3006$ πόντους. Αν θέλετε να δοκιμάσετε διαφορετικές στρατηγικές και να δείτε πως θα "έπαιζε" με διαφορετικούς παίκτες παραθέτουμε ένα ενδεικτικό (όχι το τελικό!) script "διαιτητή" στο ακόλουθο URL: https://github.com/progintro/data/blob/main/scripts/referee.py που μπορεί να τρέξει διαφορετικές εκδοχές του προγράμματός σας για έναν αριθμό γύρων και να σας δώσει το score του πρώτου παίκτη. Για παράδειγμα:

```
# Download referee
$ curl -O https://raw.githubusercontent.com/progintro/data/main/scripts/referee.py
# Check what's my score against the `coop` implementation
$ python3 referee.py ./smart_coop ./coop 1000
3016
```

Στο αρχείο README.md πρέπει να προσθέσετε οποιεσδήποτε παρατηρήσεις σας κατά την διεκπεραίωση της άσκησης. Ο κώδικας απαιτείται να είναι καλά τεκμηριωμένος με σχόλια καθώς αυτό θα είναι μέρος της βαθμολόγησης. Για τις υποβολές με την καλύτερη απόδοση (αυτές που θα συγκεντρώσουν τους περισσότερους βαθμούς σε παιχνίδια με όλες τις άλλες υποβολές) θα υπάρχει bonus βαθμολογία και συγκεκριμένα: η πρώτη υποβολή θα λάβει +100%, η δεύτερη +70% και η τρίτη +40%. Σε περίπτωση ισοβαθμιών, θα ελέγξουμε την τεκμηρίωση των εργασιών. Αν δεν καταστεί δυνατό να λύσουμε τις ισοβαθμίες, μπορεί να επιλέξουμε περισσότερες υποβολές. Αν το επιτρέψει ο χρόνος, θα ζητήσουμε να γίνει μια mini παρουσίαση των πιο αποδοτικών λύσεων.

Αναφορές

[1] Οργανισμός για το μάθημα (GitHub progintro) . https://github.com/progintro.

[2] Πρόσκληση για Εργασία 2 . https://classroom.github.com/a/lPiOQjOl.

[3] Algorithmic Game Theory. https://en.wikipedia.org/wiki/Algorithmic_game_theory.

[4] DNA. https://en.wikipedia.org/wiki/DNA.

[5] DNA matching. https://www.ancestry.com/cs/dna-help/matches/dna-matching.

[6] DNA sequencing. https://en.wikipedia.org/wiki/DNA_sequencing.

[7] Even photoshop has limits! https://medium.com/@chiaracoetzee/maximum-resolution-of-bmp-image-file-8c729b3f833a.

[8] Game Theory. https://en.wikipedia.org/wiki/Game_theory.

[9] Prisoner's Dilemma. https://en.wikipedia.org/wiki/Prisoner%27s_dilemma.

[10] The BMP file format. https://en.wikipedia.org/wiki/BMP_file_format.

[11] Uno. https://en.wikipedia.org/wiki/Uno_(card_game).