



ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

Τμήμα Πληροφορικής

ΕΠΛ 232 – Προγραμματιστικές Τεχνικές και Εργαλεία

Άσκηση 4: Ομαδική Συγγραφή Βιβλιοθήκης (.a) και Πελάτη για Επεξεργασία Εικόνων Bitmap

Διδάσκων: Ανδρέας Αριστείδου

Υπεύθυνοι Άσκησης: Παύλος Αντωνίου και Πύρρος Μπράτσкас

Ημερομηνία Ανάθεσης: Παρασκευή, 8/11/2019

Ημερομηνία Παράδοσης: Παρασκευή, 29/11/2019, ώρα 23:59
(ο κώδικας να υποβληθεί σε ένα zip στο Moodle)

I. Στόχος Άσκησης

Στόχος αυτής της άσκησης είναι να σας επιτρέψει να βάλετε μαζί όλες τις γνώσεις και αρχές που αποκτήσατε μέσω του ΕΠΛ232. Το ειδικότερο θέμα της εργασίας αφορά τη συγγραφή μιας **βιβλιοθήκης επεξεργασίας εικόνων** τύπου BMP (Microsoft Bitmap) και η παρουσίαση των δυνατοτήτων της μέσω της υλοποίησης ενός **πελάτη**. Η υλοποίηση θα γίνει σε ομάδες των δυο (2) ατόμων. Για την υλοποίηση της άσκησης θα χρειαστεί να χρησιμοποιήσετε τα ακόλουθα στοιχεία:

Νέα Στοιχεία:

1. **Βιβλιοθήκη:** Καλείστε να υλοποιήσετε κάθε μια από τις λειτουργίες της βιβλιοθήκης, οι οποίες περιγράφονται στην ενότητα V αυτής της εκφώνησης, ως ένα ξεχωριστό module (.c) παρέχοντας τα πρότυπα συναρτήσεων σε ένα ενιαίο αρχείο κεφαλίδας (π.χ., δείτε σχετικό παράδειγμα string.h στη διάλεξη 16). Νοείται ότι η βιβλιοθήκη σας δύναται να έχει όσα άλλα modules (ζεύγη .c/.h) κρίνεται σκόπιμα. Εφαρμόστε το κατάλληλο επίπεδο απόκρυψης πληροφοριών με χρήση PRIVATE ή PUBLIC.
2. **Πελάτης:** Καλείστε να υλοποιήσετε ένα πελάτη ο οποίος θα συνδέεται στατικά με τη βιβλιοθήκη για να παρουσιάζει τις λειτουργίες της. Ο πελάτης θα ήταν καλό να αποτελείται από ένα μονάχα .c αρχείο (όλες οι υπόλοιπες λειτουργίες να ενσωματωθούν στην βιβλιοθήκη).
3. **Σχεδίαση Προγράμματος:** Προτρέπεται όπως η βιβλιοθήκη σας σχεδιαστεί από πάνω προς τα κάτω, αποφασίζοντας δηλαδή τα αρχεία κεφαλίδας συλλογικά και υλοποιώντας κάθε συνάρτηση ατομικά. Η ορθότητα κάθε συνάρτησης της βιβλιοθήκης (το οποίο ονομάζεται module στη περίπτωση μας), θα πρέπει να ελεγχθεί και από τα δυο μέλη της ομάδας αλλά θα πρέπει να υλοποιείται από ένα μόνο μέλος της ομάδας (του οποίου το όνομα θα τοποθετηθεί και ως Author στο εν λόγω αρχείο).
4. **Έλεγχος Προγράμματος:** Το τελικό λογισμικό σας θα πρέπει να ελεγχθεί στατικά (κατά τη μεταγλώττιση και με τον debugger) αλλά και δυναμικά (με valgrind για διαρροή μνήμης, profiler gprof για εύρεση συναρτήσεων που πρέπει να βελτιστοποιηθούν). Νοείται ότι κάθε module του προγράμματος σας θα συνοδεύεται από τους σχετικούς οδηγούς χρήσης για να γίνει το σχετικό white-box testing.
5. **GPL:** Για σκοπούς εξοικείωσης σας με την ορολογία του λογισμικού ανοικτού πηγαίου κώδικα, η βιβλιοθήκη σας θα πρέπει να κάνει χρήση του GPL προοιμίου (preamble) σε κάθε αρχείο πηγαίου κώδικα της βιβλιοθήκης. Ο πελάτης θα πρέπει να ανταποκρίνεται με το προοίμιο εάν δε δοθούν ορίσματα (βλέπε διάλεξη 18).
6. **Αξιολόγηση:** Η αξιολόγηση της άσκησης θα στηριχτεί στα αναλυτικά κριτήρια που θα παρουσιαστούν στο τέλος αυτής της εκφώνησης. Μεταξύ άλλων, βασικός στόχος της

άσκησης είναι να αξιολογηθεί η δομή και οργάνωση της βιβλιοθήκης σας αλλά και επίδοση του συνολικού προγράμματος σας (π.χ., πόση ώρα θα χρειαστεί να διεκπεραιώσει μια ακολουθία από εντολές; Πόσο αποδοτικά χρησιμοποιείται η μνήμη, κτλ.)

Στοιχεία από Προηγούμενες Εργασίες:

1. Το πρόγραμμα πρέπει να μεταγλωττίζεται τόσο μέσω του eCclipse όσο και μέσω της γραμμής εντολής με το makefile. **Κάθε αντικείμενο (module)** πρέπει να συμπεριλαμβάνει τους **σχετικό οδηγό χρήσης (driver functions)**.
2. **Σχόλια** και οδηγό σχολίων με χρήση του **doxygen** αλλά και διάγραμμα εξαρτήσεων αντικειμένων με χρήση του graphviz.

Οι λειτουργίες της βιβλιοθήκης σας και το αναμενόμενο αποτέλεσμα του πελάτη περιγράφονται αναλυτικότερα στην συνέχεια, αφού επεξηγηθεί πρώτα το πρόβλημα.

II. Εισαγωγή

Γνωρίζουμε ότι τα αρχεία εικόνων σε ένα υπολογιστή (π.χ., jpeg, tiff, gif, bmp, png, κτλ.) αποθηκεύουν την πληροφορία σε δυαδική μορφή (παρά σε αρχεία χαρακτήρων ASCII που είδατε μέχρι στιγμής). Τα αρχεία με κατάληξη **.BMP** (ή **.DIB – Device Independent Bitmap**), είναι μια ειδική κατηγορία εικόνων που αναπτύχθηκε από την Microsoft. Η απλή εσωτερική δομή των αρχείων αυτών τα καθιστούν κατάλληλα για την διδασκαλία βασικών αρχών που έχετε διδαχθεί στα πλαίσια του μαθήματος μας.

Γνωρίζουμε ότι κάθε εικόνα αποτελείται από ένα αριθμό *εικονοστοιχείων (pixels)*. Θεωρήστε για παράδειγμα τον 3x4 πίνακα από pixels της εικόνας 1. Αν δώσουμε κάποιο χρώμα σε κάθε τετράγωνο τότε θα δημιουργηθεί κάποια εικόνα. Αν είχαμε ένα μεγαλύτερο πίνακα και μια μεγάλη παλέτα χρωμάτων, τότε θα μπορούσαμε να φτιάξουμε μια εικόνα όπως αυτή που παρουσιάζεται στην Εικόνα 2.



Εικόνα 1



Εικόνα 2

Βασικό χαρακτηριστικό μιας εικόνας είναι το *βάθος χρώματος (color depth)*. Με τον όρο αυτό αναφερόμαστε στο πλήθος των διαφορετικών χρωμάτων από τα οποία μπορούν να επιλεγούν τα χρώματα ενός pixel. Οι τυπικές περιπτώσεις είναι οι εξής:

Βάθος χρώματος	Πλήθος χρωμάτων
1 bit	2 διαθέσιμα χρώματα (2^1)
4 bits	16 διαθέσιμα χρώματα (2^4)
8 bits ή 1 byte	256 διαθέσιμα χρώματα (2^8)
16 bits ή 2 bytes	65.536 χρώματα (2^{16})
24 bits ή 3 bytes	16.777.216 χρώματα (2^{24})
32 bits ή 4 bytes	4.294.967.296 χρώματα (2^{32})

Το μέγεθος ενός αρχείου λοιπόν εξαρτάται τόσο από το πλήθος των pixels όσο και από τα bytes τα οποία απαιτούνται για να περιγραφεί το χρώμα του κάθε pixel. Έτσι μια εικόνα 800x600 με βάθος χρώματος 24bit, έχει 800x600 pixels και κάθε pixel χρησιμοποιεί 3 bytes για να καθορίσει το χρώμα του, επομένως η εικόνα αυτή στη μνήμη (και στο δίσκο) θα χρησιμοποιεί τουλάχιστον $800 * 600 * 3 \text{ bytes} = 1.37\text{MB}$. Το “τουλάχιστον” δηλώνει ότι πέραν από τα χρήσιμα δεδομένα (data) της εικόνας, μαζί με την εικόνα αποθηκεύεται επίσης η επιπλέον πληροφορία της κεφαλίδας (header). Η κεφαλίδα αποθηκεύει διάφορες μέτα-πληροφορίες οι οποίες χαρακτηρίζουν τα pixels τα οποία ακολουθούν.

Για παράδειγμα το αρχείο της εικόνας 1 θα αποθηκευτεί στη δευτερεύουσα μνήμη στη μορφή που υποδεικνύεται στην Εικόνα 3 (το DATA είναι ο Πίνακας της εικόνας 1, από κάτω-αριστερά προς πάνω-δεξιά, γραμμή-γραμμή), ενώ τα σκιασμένα τετράγωνα υποδηλώνουν την κεφαλίδα της εικόνας.



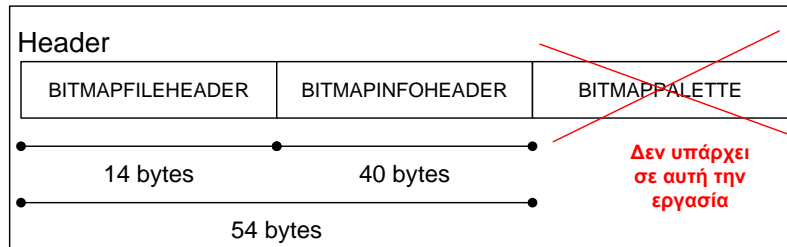
Όπως βλέπουμε, τα pixels (δηλ., το DATA) ακολουθούν τις πληροφορίες του HEADER. Όλοι οι τύποι δυαδικών αρχείων (.doc, .jpg,...) έχουν κάποιο είδος header. Συνεπώς με την ολοκλήρωση αυτής της εργασίας θα έχετε εκτιμήσει πολύ καλά τι χρειάζεται για να επεξεργαστείτε πραγματικά αρχεία δεδομένων σε ένα πρόγραμμα.

III. Δομή του Header σε Αρχεία BMP

Ας δούμε τώρα αναλυτικότερα το Header μέρος ενός BMP αρχείου. Όπως φαίνεται στο σχήμα της Εικόνας 4, το Header διαιρείται σε 3 μέρη, κάθε ένα εκ' των οποίων περιέχει κάποια συνοδευτική πληροφορία για τη συγκεκριμένη εικόνα (μέτα-πληροφορία). Τα τρία μέρη BITMAPFILEHEADER, BITMAPINFOHEADER και BITMAPPALETTE παρουσιάζονται αναλυτικότερα στη συνέχεια. Το BITMAPFILEHEADER περιέχει πληροφορίες για το **αρχείο** (π.χ., το μέγεθος του αρχείου σε bytes), το BITMAPINFOHEADER πληροφορίες για την **εικόνα** (π.χ., χρώμα, μέγεθος, κτλ) και το BITMAPPALETTE περιέχει ένα **πίνακα χρωμάτων**.

Σε αυτή την εργασία θα μας απασχολήσουν μόνο τα πρώτα δυο κομμάτια του Header ενώ το **BITMAPPALETTE θα το αγνοήσουμε**. Αυτό διότι το BITMAPPALETTE υπάρχει μόνο στις εικόνες με βάθος χρώματος μέχρι 8-bits. Οι εικόνες με βάθος χρώματος 16 bits και μεγαλύτερο δεν έχουν BITMAPPALETTE. Στις εικόνες λοιπόν με λίγα χρώματα, το BITMAPPALETTE περιγράφει ποια χρώματα χρησιμοποιεί η εικόνα (π.χ., σε μια εικόνα με 256 χρώματα, θα πρέπει να δηλώσουμε ποια είναι αυτά τα 256 χρώματα από το σύνολο των χρωμάτων που μπορεί να απεικονίσει ο υπολογιστής, π.χ., το byte έχει τα ακόλουθα bits R,R,R,G,G,G,B,B, όπου R=Red, G=Green και B=Blue). **Στην παρούσα εργασία θα ασχοληθούμε μόνο με εικόνες βάθους χρώματος 24 bits στις οποίες η κατανομή χρωμάτων είναι πάντα η ίδια για το R, G, B, δηλαδή 3 bytes συνολικά.**

Τα BITMAPFILEHEADER και BITMAPINFOHEADER έχουν συνολικό μέγεθος 54 bytes, όπως φαίνεται και στο παρακάτω σχήμα.

Εικόνα 4

Οι επόμενοι πίνακες περιγράφουν αναλυτικά τα πεδία των headers. Εσείς θα κληθείτε να επεξεργαστείτε τα πεδία αυτά για να υλοποιήσετε τα ζητούμενα της άσκησης αυτής. Ειδικότερα θα πρέπει να ορίσετε τις κατάλληλες δομές (struct) λαμβάνοντας υπόψη θέματα ευθυγράμμισης μνήμης. Για ευκολία αναφοράς καλείστε να χρησιμοποιήσετε τις ακόλουθες δηλώσεις:

```
typedef unsigned char byte;
typedef unsigned short int word;
typedef unsigned int dword;
```

Τα σκιασμένα πεδία δηλώνουν τα πεδία τα οποία μάλλον δεν θα χρειαστείτε να επεξεργαστείτε (εάν και πρέπει να είναι μέρος των δομών σας).

BITMAPFILEHEADER:

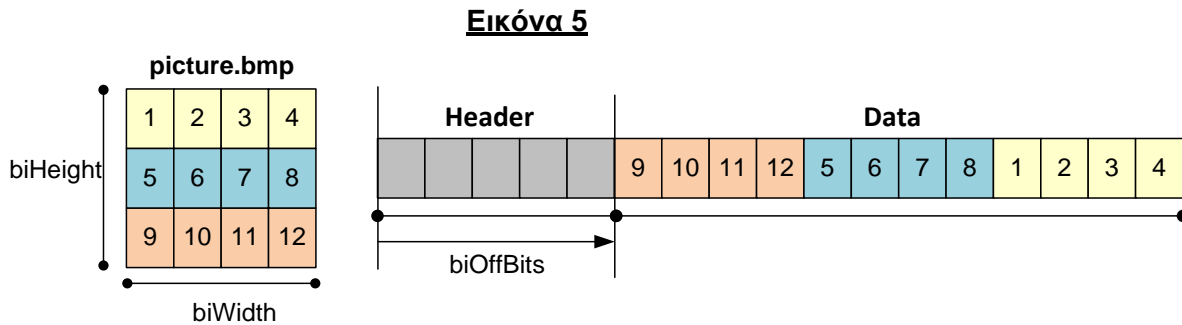
Bytes	Όνομα	Χρήση
1	bfType1	Must always be set to 'BM' to declare that this is a .bmp file (i.e., bfType1='B' and bfType2='M'. Based on these two fields you can identify if this is Bitmap file or not.
1	bfType2	
4	bfSize	Specifies the size of the file (including padding) in bytes
2	bfReserved1	Usually set to zero.
2	bfReserved2	Usually set to zero.
4	bfOffBits	Specifies the offset from the beginning of the file to the bitmap data.

BITMAPINFOHEADER:

Bytes	Όνομα	Χρήση
4	biSize	Specifies the size of the BITMAPINFOHEADER structure, in bytes.
4	biWidth	Specifies the width of the image, in pixels.
4	biHeight	Specifies the height of the image, in pixels.
2	biPlanes	Specifies the number of planes of the target device, usually set to zero.
2	biBitCount	Specifies the number of bits per pixel.
4	biCompression	Specifies the type of compression, usually set to zero (no compression). Need to provide an error if image is compressed.
4	biSizeImage	Specifies the size of the image data, in bytes. If there is no compression, it is valid to set this member to zero.
4	biXPelsPerMeter	Specifies the horizontal pixels per meter on the designated target device, usually set to zero.
4	biYPelsPerMeter	Specifies the vertical pixels per meter on the designated target device, usually set to zero.
4	biClrUsed	Specifies the number of colors used in the bitmap, if set to zero the number of colors is calculated using the biBitCount member.
4	biClrImportant	Specifies the number of color that are 'important' for the bitmap, if set to zero, all colors are important.

IV. Δομή του DATA σε Αρχεία BMP

Τώρα θα δούμε πώς αποθηκεύονται τα DATA (δηλαδή τα πραγματικά pixels μιας εικόνας) μέσα στο αρχείο (σε μη-συμπίεσμενη μορφή, ενώ η συμπίεσμένη μορφή δεδομένων θα αγνοηθεί για την εργασία). Η περιοχή DATA, αποθηκεύει σειριακά τις γραμμές της εικόνας, από το **κάτω-αριστερά** άκρο της εικόνας προς το **άνω-δεξιά** άκρο. Σχηματικά, η αποθήκευση φαίνεται στο πιο κάτω σχήμα:



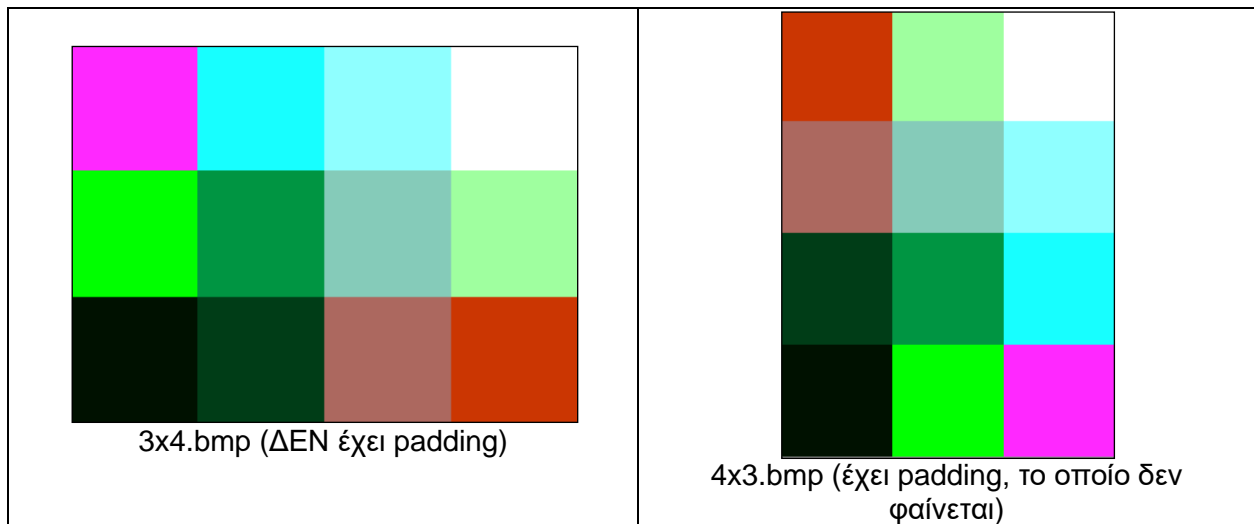
Σημειώστε ότι το **biWidth** (μέσα στο BITMAPINFOHEADER) δίνει το πλάτος μιας εικόνας ενώ το **biHeight** (μέσα στο BITMAPINFOHEADER) το ύψος της εικόνας. Επίσης το **biBitCount** (μέσα στο BITMAPINFOHEADER) δίνει το βάθος χρώματος (σε αυτή την άσκηση όλες οι εικόνες είναι 24 bits – δηλαδή κάθε pixel θα καταλαμβάνει 3 bytes). Σημειώστε επίσης ότι το πεδίο **bfOffBits** (μέσα στο BITMAPFILEHEADER) σας δίνει την ακριβή θέση (σε σχέση με την αρχή του αρχείου) απ' όπου ξεκινούν τα Data, εάν και δεν χρειάζεται μάλλον να χρησιμοποιηθεί.

Περιπλοκές

Όπως αναφέραμε, μια εικόνα έχει **biWidth** αριθμό pixels σε κάθε γραμμή. Εάν το $(biWidth * 3) \% 4$ αφήνει υπόλοιπο μηδέν, τότε κάθε γραμμή της εικόνας δεν έχει τίποτα άλλο στο τέλος της. Εναλλακτικά, εάν $(biWidth * 3) \% 4$ αφήνει υπόλοιπο διάφορο του μηδέν, τότε περιέχει επιπλέον $4 - ((3 * biWidth) \% 4)$ κενά **pixels** στο τέλος κάθε γραμμής, τα οποία ονομάζονται **padding**. Κάθε padding pixel αποτελείται από 3 κενά bytes (δηλ., έχει τη τιμή NUL – 0x00). Σημειώστε ότι τα padding pixels δεν είναι μέρος του περιεχομένου της εικόνας αλλά παρουσιάζονται στο αρχείο της εικόνας για λόγους ευθυγράμμισης που δεν θα μας απασχολήσουν.

Για να κατανοήσετε την πιο πάνω περιπλοκή θεωρήστε τις ακόλουθες δυο περιπτώσεις:

- Η πιο κάτω εικόνα **3x4.bmp (αριστερά)** έχει 4 pixels σε κάθε γραμμή. Εφόσον το $(biWidth * 3) \% 4$ αφήνει υπόλοιπο μηδέν, αυτό σημαίνει ότι το αρχείο **ΔΕΝ** θα έχει οποιοδήποτε **padding pixel** σε κάθε γραμμή.
- Η πιο κάτω εικόνα **4x3.bmp (δεξιά)** από την άλλη έχει 3 pixels σε κάθε γραμμή. Εφόσον το $(biWidth * 3) \% 4$ αφήνει υπόλοιπο ίσο με 1, αυτό σημαίνει ότι η κάθε γραμμή θα έχει $4 - ((biWidth * 3) \% 4)$, δηλαδή **1 padding pixel** (δηλαδή 3 επιπλέον bytes).




Κατανόηση Δομής Δυναμικών Αρχείων με Hexdump

Η πιο κάτω εκτέλεση της εντολής `hexdump` δείχνει το περιεχόμενο του **4x3.bmp** αρχείου σε bytes: η **πρώτη στήλη** δείχνει τη διεύθυνση της γραμμής σε δεκαεξαδική αναπαράσταση (π.χ., 00000020 σε δεκαεξαδική αναπαράσταση είναι το 32 σε δεκαδική αναπαράσταση), η **δεύτερη στήλη** δείχνει ένα-ένα τα bytes του αρχείου (για χάρη παρουσίασης κάθε γραμμή περιέχει 16 bytes) και η **τρίτη στήλη** δείχνει κάθε ένα από τα 16 bytes της γραμμής ως να ήταν ASCII χαρακτήρες (στη περίπτωση μας μόνο οι πρώτοι 2 χαρακτήρες, BM είναι πραγματικά ASCII χαρακτήρες, τα υπόλοιπα είναι ακολουθίες δυάδων `byte`, `word`, `dword`).

`$ hexdump -C 4x3.bmp`

ΣΤΗΛΗ 1	ΣΤΗΛΗ 2	ΣΤΗΛΗ 3
00000000	42 4d 68 00 00 00 00 00 00 00 36 00 00 00 28 00	BMh.....6...(.
00000010	00 00 03 00 00 00 04 00 00 00 01 00 18 00 00 00
00000020	00 00 32 00 00 00 12 0b 00 00 12 0b 00 00 00 00	..2.....
00000030	00 00 00 00 00 00 00 11 00 00 ff 00 ff 28 ff 00(..
00000040	00 00 17 3d 00 42 95 00 ff ff 17 00 00 00 5f 68	...=.B....._h
00000050	ac b9 cb 85 ff ff 8f 00 00 00 02 36 cb 9f ff 9f6....
00000060	ff ff ff 00 00 00 00 00
00000068		

Από την πιο πάνω έξοδο της `hexdump` παρατηρούμε τα ακόλουθα: **α)** τα πρώτα 54 bytes, τα οποία είναι σκιασμένα με γκριζό, για χάρη παρουσίασης, αναφέρονται στο header της εικόνας. **β)** Το header ακολουθείται από 4 γραμμές pixels, η κάθε μια εκ των οποίων γραμμές έχει 3 pixel (υπογραμμισμένα) και 1 pixel padding (3 bytes σκιασμένο και κόκκινο). Συνολικά δηλαδή, σε κάθε γραμμή υπάρχουν 16 pixels, εκ των οποίων τα χρήσιμα είναι μόνο τα 12 υπογραμμισμένα. **γ)** Το πρώτο pixel (από κάτω αριστερά προς κάτω δεξιά) της εικόνας 4x3 είναι αυτό που μοιάζει με "μαύρο" (001100), το δεύτερο αυτό που μοιάζει με "πράσινο ανοικτό" (00ff00) και το τρίτο αυτό που μοιάζει με "ροζ" (ff28ff). Παρατηρήστε ότι κάθε pixel αποτελείται από 24 bits (8bits κόκκινο, 8 bits πράσινο και 8 bits μπλέ), δηλαδή προκύπτει η πιο κάτω ακολουθία 24 bits: RRRRRRRRRGGGGGGGGBBBBBBBB.

Συμβουλή: Χρησιμοποιήστε το εργαλείο GIMP (είναι δωρεάν εάν το θέλετε για τον Η/Υ σας!), για να δείτε τις εικόνες που βρίσκονται μαζί με την εκφώνηση (για τις εικόνες 3x4 και 4x3 κάνετε zoom έτσι ώστε να μπορείτε να επιλέξετε ένα pixel και να δείτε τι ακριβώς χρώμα έχει, κάνοντας χρήση του color picker , π.χ., δείτε <http://docs.gimp.org/en/gimp-tool-color-picker.html>)

Το αρχείο εισόδου που βρίσκονται στην ιστοσελίδα περιέχουν τόσο εικόνες με padding όσο και εικόνες χωρίς padding. Βεβαιωθείτε ότι το πρόγραμμα σας εκτελείται ορθά για

όλες τις εικόνες που δίνονται. Τέλος σημειώστε ότι τα padding δεν είναι ορατά με το συμβατικό gimp viewer αλλά μόνο με το hexdump.

V. Ζητούμενα Άσκησης

Έστω ότι `bmplib.a` είναι η βιβλιοθήκη που θα δημιουργήσετε και `bmpengine.c` ο πελάτης, τότε `bmpengine` είναι το εκτελέσιμο αρχείο που παράγεται συνδέοντας το `bmplib.a` με το `bmpengine.c` κατά την μεταγλώττιση (υποδειγματικές εικόνες έχουν αναρτηθεί στο `as4-supplementary.zip`). Πιο κάτω περιγράφεται η αναμενόμενη λειτουργία της βιβλιοθήκης κάνοντας αναφορά στις λειτουργίες της μέσω υποδειγματικών εκτελέσεων του πελάτη.

A) Πελάτης

Η γενική μορφή εκτέλεσης του πελάτη είναι η ακόλουθη:

```
$bmpengine <-option> image1.bmp [ image2.bmp image3.bmp ...]
```

όπου `option` σχετίζεται με μια επί μέρους λειτουργία της βιβλιοθήκης (π.χ., `-list` εκτυπώνει το header, που θα δούμε σε λίγο) και στη συνέχεια ακολουθεί ένας απροσδιόριστος αριθμός από ονόματα αρχείων.

Ενδεχόμενα Λάθη:

- **Δεν ορίζεται option ή δεν δίνεται το όνομα τουλάχιστο μιας εικόνας:** Πρέπει να εκτυπώνεται το GPL προοίμιο και στη συνέχεια να δίνεται το σχετικό μήνυμα λάθους.
- **Μια εικόνα δεν είναι .bmp ή δεν είναι 24-bit ή είναι συμπιεσμένη:** Η εικόνα πρέπει να αγνοείται (η αναγνώριση να γίνει από το `bfType` και όχι από το extension του αρχείου) και να τυπώνεται το GPL προοίμιο.

B) Βιβλιοθήκη

Λειτουργία 1: Εξαγωγή Μέτα-Πληροφοριών

Το όρισμα `-list` αφορά τη εκτύπωση των στοιχείων του header μιας προσδιορισμένης εικόνας σε μορφή που ακολουθεί.

```
$./bmpengine -list 4x3.bmp image2.bmp
```

```
BITMAP_FILE_HEADER
=====
```

```
bfType: BM
bfSize: 104
bfReserved1: 0
bfReserved2: 0
bfOffBits: 54
```

```
BITMAP_INFO_HEADER
=====
```

```
biSize: 40
biWidth: 3
biHeight: 4
biPlanes: 1
biBitCount: 24
biCompression: 0
biSizeImage: 50
biXPelsPerMeter: 2834
biYPelsPerMeter: 2834
biClrUsed: 0
biClrImportant: 0
```

```
*****
```



```

BITMAP_FILE_HEADER
=====
bfType: BM
bfSize: 322854
bfReserved1: 0
bfReserved2: 0
bfOffBits: 54

BITMAP_INFO_HEADER
=====
biSize: 40
biWidth: 400
biHeight: 269
biPlanes: 1
biBitCount: 24
biCompression: 0
biSizeImage: 322800
biXPelsPerMeter: 0
biYPelsPerMeter: 0
biClrUsed: 0
biClrImportant: 0

```

Λειτουργία 2: Περιστροφή Εικόνας βάσει Οριζόντιου Άξονα (Horizontal Flip)

Το όρισμα `-hflip` περιστρέφει την εικόνα βάσει του οριζόντιου άξονα. Το αποτέλεσμα αποθηκεύεται σε νέο αρχείο με όνομα `hflip-image1.bmp`, όπου `image1.bmp` είναι το αρχικό όνομα της εικόνας. Σημειώστε ότι ο χρήστης έχει και πάλι τη δυνατότητα να εισάγει ένα απροσδιόριστο αριθμό από ονόματα αρχείων όπως και στην επιλογή `list`. Σημειώστε ότι για να επιτευχθεί η στροφή αρκεί να κάνετε `swar` τις γραμμές της εικόνας με μια συγκεκριμένη λογική και δεν χρειάζεται να κάνετε οποιουδήποτε μαθηματικούς μετασχηματισμούς. Παρακάτω φαίνεται ένα παράδειγμα της στροφής 3 εικόνων:

```
$bmpengine -hflip image1.bmp image7.bmp image3x4.bmp
```

image1.bmp



hflip-image1.bmp



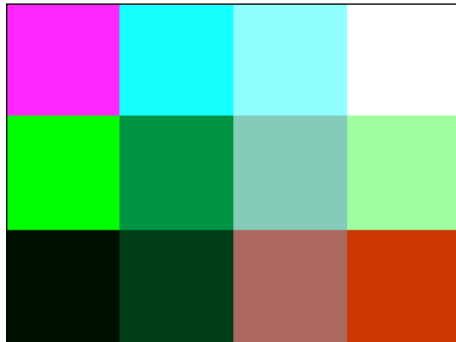
image7.bmp



hflip-image7.bmp



image3x4.bmp



hflip-image3x4.bmp



Λειτουργία 3: Περιστροφή Εικόνας βάσει Κατακόρυφου Άξονα (Vertical Flip)

Το όρισμα `-vflip` περιστρέφει την εικόνα βάσει του κατακόρυφου άξονα. Το αποτέλεσμα αποθηκεύεται σε νέο αρχείο με όνομα `vflip-image1.bmp`, όπου `image1.bmp` είναι το αρχικό όνομα της εικόνας. Σημειώστε ότι για να επιτευχθεί η στροφή αρκεί να κάνετε `swap` τις στήλες της εικόνας με μια συγκεκριμένη λογική και δεν χρειάζεται να κάνετε οποιοσδήποτε μαθηματικούς μετασχηματισμούς. **Ιδιαίτερη προσοχή πρέπει να δοθεί στο padding, εάν υπάρχει!** Σημειώστε ότι ο χρήστης έχει και πάλι τη δυνατότητα να εισάγει ένα απροσδιόριστο αριθμό από ονόματα αρχείων όπως και στην επιλογή `list`. Παρακάτω φαίνεται ένα παράδειγμα περιστροφής 2 εικόνων:

```
$bmpengine -vflip image1.bmp image7.bmp
```

image1.bmp



vflip-image1.bmp



image7.bmp



vflip-image7.bmp



Λειτουργία 4: Περιστροφή εικόνας κατά 90 Μοίρες Δεξιόστροφα (Right Rotate)

Το όρισμα `-right90` περιστρέφει τις εικόνες που δίνονται σαν ορίσματα κατά 90 μοίρες δεξιόστροφα, σύμφωνα με τη φορά περιστροφής των δεικτών του ρολογιού. Το αποτέλεσμα αποθηκεύεται σε νέο αρχείο με όνομα `right-image1.bmp`, όπου `image1.bmp` είναι το αρχικό όνομα της εικόνας. Σημειώστε ότι ο χρήστης έχει και πάλι τη δυνατότητα να εισάγει ένα απροσδιόριστο αριθμό από ονόματα αρχείων όπως και στην επιλογή `list`. Παρακάτω φαίνεται ένα παράδειγμα αντιστροφής 1 εικόνας:

```
$bmpengine -right90 image1.bmp
```

image1.bmp



right90-image1.bmp



Υπόδειξη: Εάν η αρχική εικόνα **Image₁** έχει διαστάσεις $N \times M$ pixels, η περιστραμμένη εικόνα **Image₂** θα έχει διαστάσεις $M \times N$ pixels. Το τυχαίο pixel **Image₂(j,i)** της περιστραμμένης εικόνας ισούται κατά περίπτωση με:

$$\mathbf{Image}_2(j,i) = \begin{cases} \mathbf{Image}_1(i, M - j), & \text{δεξιόστροφα} \\ \mathbf{Image}_1(N - i, j), & \text{αριστερόστροφα} \end{cases}$$

όπου το $j \in [0..M-1]$ και $i \in [0..N-1]$. Προσοχή χρειάζεται στον υπολογισμό των padding bytes, τα οποία στη νέα εικόνα μπορεί να είναι διαφορετικά. Αυτό έχει ως αποτέλεσμα οι δύο εικόνες να διαφέρουν ενδεχομένως

λίγο στο μέγεθος τους (χρειάζεται λοιπόν ενημέρωση και των πεδίων `biSizeImage` και `bfSize` στα headers της νέας εικόνας). Η αριστερόστροφη περιστροφή της εικόνας γίνεται στην επόμενη λειτουργία 5.

Λειτουργία 5: Περιστροφή Εικόνας κατά 90 μοίρες Αριστερόστροφα (Left Rotate)

Το όρισμα `-left90` περιστρέφει τις εικόνες που δίνονται σαν ορίσματα κατά 90 μοίρες αριστερόστροφα, αντίθετα με τη φορά περιστροφής των δεικτών του ρολογιού. Το αποτέλεσμα αποθηκεύεται σε νέο αρχείο με όνομα `left-image1.bmp`, όπου `image1.bmp` είναι το αρχικό όνομα της εικόνας. Σημειώστε ότι ο χρήστης έχει και πάλι τη δυνατότητα να εισάγει ένα απροσδιόριστο αριθμό από ονόματα αρχείων όπως και στην επιλογή `list`. Παρακάτω φαίνεται ένα παράδειγμα αντιστροφής 1 εικόνας:

```
$bmpengine -left90 image1.bmp
```

image1.bmp



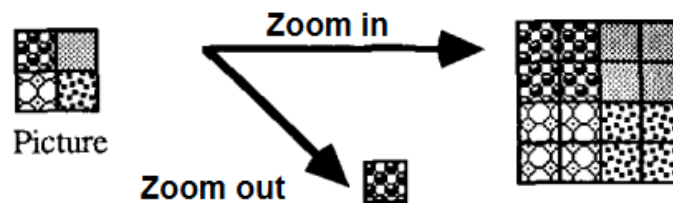
left90-image1.bmp



Λειτουργία 6: Μεγέθυνση Εικόνας (Zoom in)

```
$bmpengine -zoomin image1.bmp [ image2.bmp image3.bmp ...]
```

Το όρισμα `-zoomin` μεγεθύνει τις εικόνες που δίνονται στο 200% απλά αντιγράφοντας κάθε pixel σε μια γειτονιά 4 pixel όπως φαίνεται στο πιο κάτω σχήμα. Με άλλα λόγια, η λειτουργία `zoom in` δημιουργεί για κάθε ένα pixel της αρχικής εικόνας άλλα 3 pixel ακριβώς τα ίδια. Ιδιαίτερη προσοχή θα πρέπει να επιδείξετε στο γεγονός ότι κάποια πεδία της κεφαλίδας αλλάζουν τιμή.



Παρακάτω φαίνεται ένα παράδειγμα μεγέθυνσης μιας εικόνας:

```
$bmpengine -zoomin image1.bmp
```


image1.bmp



zoomin-image1.bmp



Λειτουργία 7: Σμίκρυνση Εικόνας (Zoom out)

```
$bmpengine -zoomout image1.bmp [ image2.bmp image3.bmp ...]
```

Το όρισμα `-zoomout` σμικρύνει τις εικόνες που δίνονται στο 50%, αφαιρώντας τρία pixel από κάθε 4 γειτονικά pixel όπως φαίνεται στο πιο πάνω σχήμα. Με άλλα λόγια, ξεκινώντας από πάνω αριστερά της εικόνας, για κάθε 2x2 pixels μένει μόνο το πάνω αριστερά. Ιδιαίτερη προσοχή θα πρέπει να επιδείξετε στο γεγονός ότι κάποια πεδία της κεφαλίδας αλλάζουν τιμή. Για να διασφαλιστεί η ακεραιότητα σμίκρυνσης, μπορείτε να σμικρύνεται εικόνα η οποία έχει μεγεθυνθεί μέσω της προηγούμενης εντολής. Τα ονόματα των νέων εικόνων που δημιουργούνται πρέπει να έχουν το πρόθεμα `zoomout-`

Λειτουργία 8: Φίλτρο Grayscale

```
$bmpengine -gray image1.bmp [ image2.bmp image3.bmp ...]
```

Το όρισμα `-gray` εφαρμόζει το φίλτρο grayscale το οποίο μετατρέπει το χρώμα του κάθε pixel σε αποχρώσεις του γκριζου. Για να γίνει αυτό κατορθωτό θα πρέπει κάθε pixel να μετατρέπεται με την ακόλουθη λογική: Κάθε pixel καταλαμβάνει 3 bytes (εφόσον χρησιμοποιείται 24bit βάθος χρώματος). Ειδικότερα, τα bytes αυτά κατανέμονται ως ακολούθως: red=1 byte, green=1 byte, blue=1 byte.

Τα bytes αυτά πολλαπλασιάζονται χρησιμοποιώντας την NTSC (National Television System Committee) εξίσωση: $0.299 \cdot \text{red} + 0.587 \cdot \text{green} + 0.114 \cdot \text{blue}$. Το **στρογγυλοποιημένο** αποτέλεσμα της NTSC εξίσωσης πάνω σε κάθε επί μέρους 3-byte RGB ακολουθία του αρχείου θα ονομάζεται απόλυτη φωτεινότητα (luminance). Για παράδειγμα για ένα pixel RGB=(9,90,160), το luminance είναι $0.299 \cdot 9 + 0.587 \cdot 90 + 0.114 \cdot 160 = 73,761$. Συνεπώς, η γκριζα έκδοση του pixel πρέπει να είναι **(74,74,74)**.

Εάν επαναλάβουμε την πιο πάνω εξίσωση για όλα τα pixel μιας εικόνας θα έχουμε το ακόλουθο αποτέλεσμα.

image1.bmp



gray-image1.bmp



Λειτουργία 9: Φίλτρο Όξυνσης Εικόνας (Sharpen)

```
$bmpengine -sharpen image1.bmp [ image2.bmp image3.bmp ...]
```

Η διαδικασία όξυνσης (sharpening) είναι μια διαδικασία επεξεργασίας της εικόνας με σκοπό να οξύνουμε (τονίσουμε) τις ακμές (όρια) των αντικειμένων (περιοχών) που απεικονίζονται σε μια εικόνα και ανήκει στην κατηγορία των χωρικών φίλτρων (spatial filters). Με τον όρο χωρικά φίλτρα αναφερόμαστε σε μετασχηματισμούς που εφαρμόζονται πάνω στα **pixel** της εικόνας με τη χρήση μιας μάσκας. Η μάσκα είναι ένας πίνακας δύο διαστάσεων με κάποιες αριθμητικές τιμές (βάρη).

Έστω ότι έχουμε την εικόνα της οποίας το pixel στην θέση (i,j) είναι το $z(i,j)$. Αν εφαρμόσουμε την 3x3 μάσκα M:

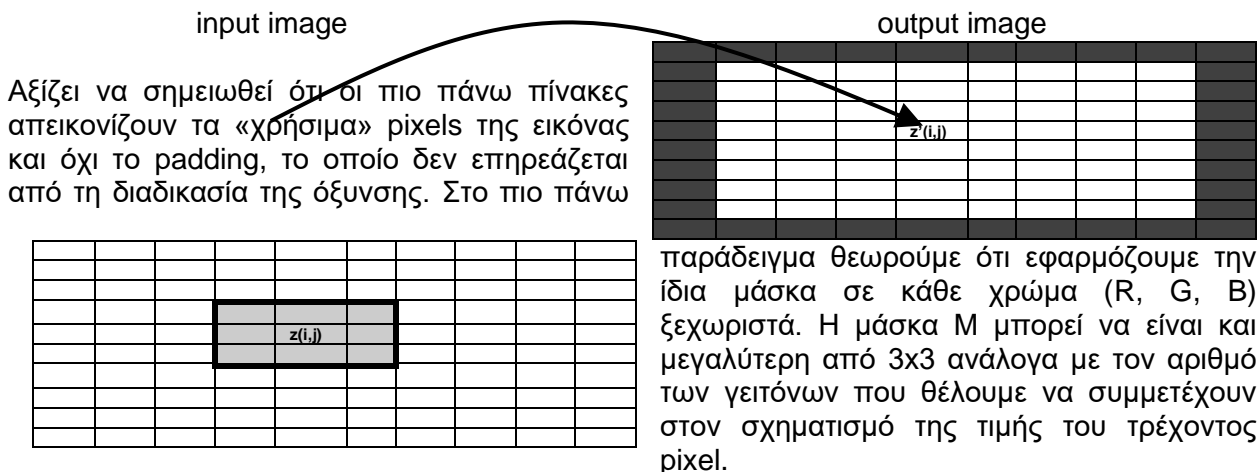
m1	m2	m3
m4	m5	m6
m7	m8	m9

στην εικόνα μας τότε το pixel (η κάθε συνιστώσα του pixel ξεχωριστά) στην θέση (i,j) θα πάρει την τιμή :

$$z'(i,j)_{\text{RED}} = m1 * z(i-1,j-1)_{\text{RED}} + m2 * z(i-1,j)_{\text{RED}} + m3 * z(i-1,j+1)_{\text{RED}} + \\ m4 * z(i,j-1)_{\text{RED}} + m5 * z(i,j)_{\text{RED}} + m6 * z(i,j+1)_{\text{RED}} + \\ m7 * z(i+1,j-1)_{\text{RED}} + m8 * z(i+1,j)_{\text{RED}} + m9 * z(i+1,j+1)_{\text{RED}}$$

$$z'(i,j)_{\text{GREEN}} = m1 * z(i-1,j-1)_{\text{GREEN}} + m2 * z(i-1,j)_{\text{GREEN}} + m3 * z(i-1,j+1)_{\text{GREEN}} + \\ m4 * z(i,j-1)_{\text{GREEN}} + m5 * z(i,j)_{\text{GREEN}} + m6 * z(i,j+1)_{\text{GREEN}} + \\ m7 * z(i+1,j-1)_{\text{GREEN}} + m8 * z(i+1,j)_{\text{GREEN}} + m9 * z(i+1,j+1)_{\text{GREEN}}$$

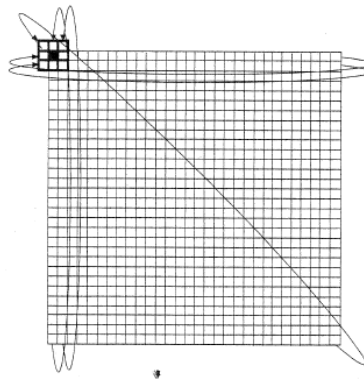
$$z'(i,j)_{\text{BLUE}} = m1 * z(i-1,j-1)_{\text{BLUE}} + m2 * z(i-1,j)_{\text{BLUE}} + m3 * z(i-1,j+1)_{\text{BLUE}} + \\ m4 * z(i,j-1)_{\text{BLUE}} + m5 * z(i,j)_{\text{BLUE}} + m6 * z(i,j+1)_{\text{BLUE}} + \\ m7 * z(i+1,j-1)_{\text{BLUE}} + m8 * z(i+1,j)_{\text{BLUE}} + m9 * z(i+1,j+1)_{\text{BLUE}}$$



Στην λειτουργία αυτή θα χρησιμοποιήσουμε την 3x3 μάσκα που δίνεται πιο κάτω:

-1	-1	-1
-1	9	-1
-1	-1	-1

Για τους υπολογισμούς των pixels στα όρια της εικόνας (με έντονο χρώμα στο output image) θα ακολουθήσετε τη στρατηγική του πιο κάτω σχήματος:



Παρακάτω φαίνεται ένα παράδειγμα όξυνσης μιας εικόνας:

```
$bmpengine -sharpen image1.bmp
```

image1.bmp



sharpen-image1.bmp



Γ) Αξιολόγηση Εργασίας

Η αξιολόγηση της άσκησης θα γίνει στο εργαστήριο (ανά ομάδα) σε ημερομηνία που θα ανακοινωθεί αργότερα. Κατά την διάρκεια της αξιολόγησης της εργασίας θα πρέπει να γίνει η επίδειξη της σχεδίασης και υλοποίησης της βιβλιοθήκης και του πελάτη σας απ' όλα τα μέλη της ομάδας. Τυχούσα παράληψη παρουσίασης της εργασίας ενδέχεται να συνεπάγεται τον μηδενισμό της εργασίας. Στοιχεία τα οποία θα ληφθούν υπόψη στην αξιολόγηση της εργασίας σας περιλαμβάνουν: ορθότητα λειτουργίας, στοιχεία επίδοσης: π.χ., ελαχιστοποίηση του *χρόνου απόκρισης* σε αιτήσεις, το οποίο ορίζεται ως το διάστημα μεταξύ της χρονικής στιγμής που γίνεται η αίτηση και της στιγμής που διεκπεραιώνεται μια λειτουργία, σχεδίαση μονάδων λογισμικού και της βιβλιοθήκης ευρύτερα.

Δ) Γενικές Οδηγίες Υποβολής

Το πρόγραμμά σας θα πρέπει να συμβαδίζει με το πρότυπο ISO C, να περιλαμβάνει εύστοχα και περιεκτικά σχόλια, να έχει καλή στοίχιση και το όνομα κάθε μεταβλητής, σταθεράς, ή συνάρτησης να είναι ενδεικτικό του ρόλου της. Να χρησιμοποιήσετε το λογισμικό τεκμηρίωσης *doxygen* έτσι ώστε να μπορούμε να μετατρέψουμε τα σχόλια του προγράμματός σας σε HTML αρχεία και να τα δούμε με ένα browser. Η συστηματική αντιμετώπιση της λύσης ενός προβλήματος περιλαμβάνει στο παρόν στάδιο τη διάσπαση του προβλήματος σε μικρότερα ανεξάρτητα προβλήματα που κατά κανόνα κωδικοποιούμε σε ξεχωριστές συναρτήσεις. Για αυτό τον λόγο σας καλούμε να κάνετε χρήση συναρτήσεων και άλλων τεχνικών δομημένου προγραμματισμού που διδαχτήκατε στο ΕΠΛ131. Επίσης, σας θυμίσουμε ότι κατά την διάρκεια της εκτέλεσης του προγράμματος σας αυτό θα πρέπει να δίνει τα κατάλληλα μηνύματα σε περίπτωση λάθους. **Τέλος το πρόγραμμά σας θα πρέπει να μεταγλωττίζεται στις μηχανές του εργαστηρίου.**

Παρακαλώ όπως μελετηθούν ξανά οι κανόνες υποβολής εργασιών όπως αυτοί ορίζονται στο συμβόλαιο του μαθήματος. Επίσης να ακολουθήσετε τα πιο κάτω βήματα όταν υποβάλετε την άσκηση σας στο Moodle:

1. Δημιουργήστε ένα κατάλογο με το όνομά σας π.χ., PavlosAntoniou/ χωρίς να αφήνετε κενά στο όνομα του καταλόγου.
2. Βάλτε μέσα στον κατάλογο αυτό όλα τα αρχεία της εργασίας (κώδικας (αρχεία .c και .h), doxygen configuration file, README.dox, βιβλιοθήκη .a) που πρέπει να υποβάλετε.
3. Συμπιέστε (zip) τον κατάλογο (και όχι τα αρχεία ξεχωριστά) χρησιμοποιώντας την εντολή `zip -r PavlosAntoniou.zip PavlosAntoniou/*`
4. **Βεβαιωθείτε ότι κάνατε σωστά τα τρία προηγούμενα βήματα**
5. Ανεβάστε στο Moodle το συμπιεσμένο αρχείο π.χ., PavlosAntoniou.zip ENA ZIP file ανά ομάδα (μόνο ένα μέλος της ομάδας να ανεβάσει)

Ε) Κριτήρια αξιολόγησης:

Υλοποίηση Πελάτη	10
Λειτουργία 1: Εκτύπωση list	6
Λειτουργία 2: Μετασχηματισμός hflip	7
Λειτουργία 3: Μετασχηματισμός vflip	7
Λειτουργία 4: Μετασχηματισμός right90	10
Λειτουργία 5: Μετασχηματισμός left90	7
Λειτουργία 6 Μετασχηματισμός zoomin	10
Λειτουργία 7 Μετασχηματισμός zoomout	7
Λειτουργία 8 Φίλτρο grayscale	6
Λειτουργία 9 Φίλτρο sharpen	10
Γενική εικόνα (στοιχισμένος και ευανάγνωστος κώδικας, εύστοχα ονόματα μεταβλητών, σταθερών και συναρτήσεων, σχολιασμός doxygen, αρχείο .conf, αρχείο README.dox)	8
Εκτύπωση GPL Προοιμίων εκεί που ζητείται, σωστή σχεδίαση προγράμματος με πολλαπλά αρχεία, gprof report, valgrind report (για valgrind report ανακατευθύνετε την έξοδο της εντολής valgrind σε αρχείο). Για να παράξετε τα 2 reports χρησιμοποιήστε μια οποιαδήποτε λειτουργία.	10
Παράδοση αρχείου .a	2
ΣΥΝΟΛΟ	100
(Προαιρετικό) Επιπλέον λειτουργίες οι οποίες τυγχάνουν να έχουν ενδιαφέρον, πολυπλοκότητα και σκοπό (π.χ., περιστροφή δεξιόστροφα/αριστερόστροφα βάσει αυθαίρετης γωνίας, άλλα γνωστά φίλτρα με σχετική επεξήγηση, κτλ.)	10

Καλή Επιτυχία !