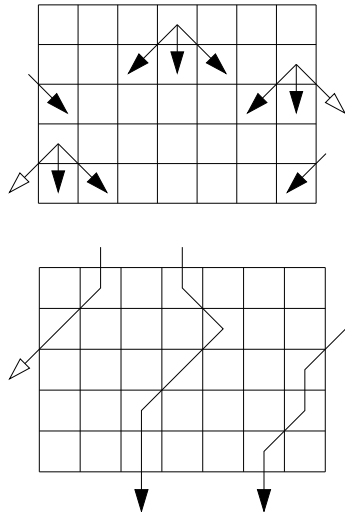


Εργασία 3

Έστω ότι έχουμε ένα ορθογώνιο πλαίσιο $n \times m$ (n γραμμές και m στήλες), το οποίο θέλουμε να διασχίσουμε από την πρώτη γραμμή προς την τελευταία. Αρχίζουμε από κάποιο τετραγωνίδιο της πρώτης γραμμής και σε κάθε βήμα μπορούμε να πηγαίνουμε σε τετραγωνίδιο της επόμενης, είτε στο ακριβώς από κάτω, είτε σε κάποιο από τα διαγώνια (κάτω αριστερά ή κάτω δεξιά). Θεωρούμε ότι στις ακραίες στήλες του ο πίνακας “αναδιπλώνεται”, οπότε αν σε μία γραμμή βρισκόμαστε στην πλέον δεξιά θέση, ως διαγώνια κάτω δεξιά αυτής της θέσης εκλαμβάνουμε τη πλέον αριστερή θέση της επόμενης γραμμής. Αντίστοιχη αναδίπλωση μπορεί να γίνει και στην αριστερή στήλη.

Οι πιθανές μεταβάσεις από ένα τετραγωνίδιο του πλαισίου σε κάποιο της επόμενης γραμμής φαίνονται στο διπλανό σχήμα (επάνω). Με αυτό τον τρόπο, μπορούμε να δημιουργήσουμε μονοπάτια διάσχισης του πλαισίου από την πρώτη προς την τελευταία γραμμή, όπως αυτά που φαίνονται στο διπλανό σχήμα (κάτω).

Ας θεωρήσουμε τώρα ότι σε κάθε τετραγωνίδιο του πλαισίου υπάρχει ένας μη αρνητικός ακέραιος αριθμός. Το ζητούμενο είναι να βρούμε ποιο είναι το μέγιστο άθροισμα αριθμών που μπορούμε να έχουμε διασχίζοντας ένα μονοπάτι, με τη λογική της προηγούμενης παραγράφου. Στα παρακάτω σχήματα φαίνονται δύο τέτοια μονοπάτια σε δύο πλαίσια 6×5 . Στο αριστερό πλαίσιο το μέγιστο άθροισμα ισούται με 44 και στο δεξιό με 49.



7	4	5	2	7
6	9	1	6	3
9	2	7	9	8
8	8	1	7	2
2	3	1	8	4
4	6	5	4	6

7	4	5	2	7
6	9	1	6	3
9	2	7	9	8
8	8	1	7	9
2	3	1	8	8
4	6	5	4	7

Αν θέλουμε να αναπτύξουμε έναν αλγόριθμο, για δεδομένο πίνακα μη αρνητικών ακεραίων $A = (a_{ij})$, διάστασης $n \times m$, που να βρίσκει το μέγιστο άθροισμα S ακολουθώντας ένα μονοπάτι από την πρώτη προς την τελευταία γραμμή, όπως περιγράφηκε νωρίτερα, μπορούμε να σκεφτούμε ως εξής:

Έστω ότι βρισκόμαστε στη θέση (i, j) του πίνακα ($1 \leq i \leq n$ και $1 \leq j \leq m$) και θέλουμε να βρούμε το μέγιστο άθροισμα S_{ij} από την πρώτη γραμμή μέσω ενός μονοπατιού μέχρι τη θέση αυτή.

- Αν βρισκόμαστε στην πρώτη γραμμή, δηλαδή $i = 1$, τότε, προφανώς, $S_{ij} = a_{ij}$.
- Αν δεν βρισκόμαστε στην πρώτη γραμμή, δηλαδή $i > 1$, τότε το μέγιστο άθροισμα S_{ij} ισούται με το άθροισμα του a_{ij} συν το μέγιστο άθροισμα που μπορούμε να επιτύχουμε μέχρι κάποιο από τα τετραγωνίδια της προηγούμενης γραμμής που είναι ακριβώς επάνω από τη θέση (i, j) ή διαγώνια αυτής, αριστερά ή δεξιά, λαμβάνοντας υπόψη και ότι ο πίνακας αναδιπλώνεται. Δηλαδή,

συμπεριλαμβάνοντας και την περίπτωση $i = 1$, έχουμε:

$$S_{ij} = \begin{cases} a_{1j} & \text{αν } i = 1 \\ a_{i1} + \max\{S_{i-1,m}, S_{i-1,1}, S_{i-1,2}\} & \text{αν } i > 1, j = 1 \\ a_{ij} + \max\{S_{i-1,j-1}, S_{i-1,j}, S_{i-1,j+1}\} & \text{αν } i > 1, 1 < j < m \\ a_{im} + \max\{S_{i-1,m-1}, S_{i-1,m}, S_{i-1,1}\} & \text{αν } i > 1, j = m \end{cases}$$

Τότε, το μέγιστο άθροισμα S που είναι δυνατόν να έχουμε ξεκινώντας από κάποιο στοιχείο της πρώτης γραμμής και καταλήγοντας στην τελευταία γραμμή ισούται με:

$$S = \max_{j=1}^m S_{n,j}$$

Πλαίσιο υλοποίησης

Στην εργασία αυτή καλείσθε να υλοποιήσετε εναλλακτικές μεθόδους επίλυσης του παραπάνω προβλήματος. Δηλαδή, δεδομένου ενός δισδιάστατου πίνακα μη αρνητικών ακεραίων, θα πρέπει να βρίσκετε βέλτιστο μονοπάτι από την πρώτη προς την τελευταία γραμμή, σύμφωνα με τους κανόνες που αναφέρθηκαν. Συγκεκριμένα, οι μέθοδοι αυτές είναι:

- Αναδρομική μέθοδος (recursive)
- Αναδρομική μέθοδος με απομνημόνευση (recursive with memoization)
- Επαναληπτική μέθοδος με δυναμικό προγραμματισμό (iterative with dynamic programming)

Όλες οι μέθοδοι θα βασισθούν στη μαθηματική προσέγγιση που περιγράφηκε προηγουμένως και θα πρέπει να υλοποιηθούν μέσω της κλήσης από τη `main()` μίας συνάρτησης με πρωτότυπο

```
void solve(int n, int m, int **board)
```

όπου `n` είναι το πλήθος των γραμμών και `m` το πλήθος των στηλών του πίνακα `board`. Εννοείται ότι κάθε συνάρτηση `solve()` θα μπορεί να καλεί άλλες συναρτήσεις, όπου το κρίνεται απαραίτητο.

Κάθε συνάρτηση `solve()` που υλοποιεί μία από τις προαναφερθείσες μεθόδους θα πρέπει να βρίσκεται σε διαφορετικό πηγαίο αρχείο `.c`. Εννοείται ότι πρέπει να έχετε δημιουργήσει και αρχείο/α επικεφαλίδας `.h`, όπου αυτά απαιτούνται. Επίσης, σε διαφορετικό πηγαίο αρχείο θα πρέπει να βρίσκεται και η συνάρτηση `main()` του προγράμματός σας. Για να κατασκευάσετε το εκάστοτε εκτελέσιμο, ανάλογα με τη μέθοδο που πρέπει να χρησιμοποιηθεί, θα πρέπει να μεταγλωττίσετε κάθε πηγαίο αρχείο στο αντίστοιχο αντικειμενικό και μετά να συνδέσετε το αντικειμενικό αρχείο της `main()` με το κατάλληλο αντικειμενικό αρχείο της μεθόδου που σας ενδιαφέρει. Ένα παράδειγμα της διαδικασίας που πρέπει να ακολουθήσετε είναι το εξής:

```
$ gcc -c -o maxsum.o maxsum.c
$ gcc -c -o maxsumrec.o maxsumrec.c
$ gcc -c -o maxsummem.o maxsummem.c
$ gcc -c -o maxsumdp.o maxsumdp.c
$ gcc -o maxsumrec maxsum.o maxsumrec.o
$ gcc -o maxsummem maxsum.o maxsummem.o
$ gcc -o maxsumdp maxsum.o maxsumdp.o
```

Αναδρομική μέθοδος (25%)

Υλοποιήστε τη συνάρτηση `solve()` που υπολογίζει με αναδρομικό τρόπο¹ και εκτυπώνει το βέλτιστο άθροισμα στον δεδομένο πίνακα από την πρώτη μέχρι την τελευταία γραμμή του (αρχείο `maxsumrec.c`), σύμφωνα με τον αναδρομικό τύπο που περιγράφηκε προηγουμένως, καθώς και κατάλληλη `main()` που θα την καλεί (αρχείο `maxsum.c`). Η `main()` να διαβάζει από την πρότυπη είσοδο τα δεδομένα του πίνακα, όπως περιγράφεται στη συνέχεια, και να καλεί την `solve()`. Στο πρόγραμμά σας δεν επιτρέπεται να ορίσετε άλλον πίνακα εκτός από αυτόν με τα δεδομένα. Όσον αφορά τη μορφή της εισόδου στο πρόγραμμα, αρχικά δίνονται οι διαστάσεις του πίνακα (πλήθος γραμμών και πλήθος στηλών) και στη συνέχεια τα δεδομένα του πίνακα κατά γραμμές.

Αν το εκτελέσιμο πρόγραμμα που θα κατασκευάσετε τελικά ονομάζεται “`maxsumrec`”, ενδεικτικές εκτελέσεις του φαίνονται στη συνέχεια.²

```
$ ./maxsumrec
```

```
6 5
```

```
7 4 5 2 7
```

```
6 9 1 6 3
```

```
9 2 7 9 8
```

```
8 8 1 7 2
```

```
2 3 1 8 4
```

```
4 6 5 4 6
```

```
Running maxsumrec
```

```
Max sum is 44
```

```
$
```

```
$ ./maxsumrec
```

```
6 5
```

```
7 4 5 2 7
```

```
6 9 1 6 3
```

```
9 2 7 9 8
```

```
8 8 1 7 9
```

```
2 3 1 8 8
```

```
4 6 5 4 7
```

```
Running maxsumrec
```

```
Max sum is 49
```

```
$
```

```
$ cat input.txt
```

```
14 16
```

7	38	8	98	85	47	31	38	49	24	86	96	15	77	32	78
96	54	0	66	55	46	31	54	22	8	54	73	61	34	22	69
72	30	67	9	29	98	99	30	75	85	26	42	14	58	20	10
65	72	77	20	70	60	26	92	21	80	18	82	66	40	51	90
22	71	51	52	21	50	82	96	87	61	38	1	19	11	64	36
83	93	8	54	53	34	46	74	67	16	57	33	56	60	24	31
31	75	83	53	78	17	1	65	78	40	19	50	3	83	38	86
76	47	92	81	33	91	56	52	7	65	86	16	77	62	47	9
89	82	14	67	99	15	85	30	7	4	32	10	39	70	49	15
69	41	96	55	32	4	7	40	21	93	8	99	55	55	60	97

¹Επισημαίνεται ότι δεν είναι απαραίτητο η ίδια η `solve()` να είναι αναδρομική. Αυτό που ζητείται είναι να λύνει το πρόβλημα με αναδρομικό τρόπο. Θα μπορούσε, για παράδειγμα, να καλεί άλλη συνάρτηση που να είναι αναδρομική.

²Το αρχείο `input.txt` μπορείτε να το βρείτε στο <http://www.di.uoa.gr/~ip/hwfiles/maxsum/input.txt>.

89	74	64	40	41	49	70	1	5	54	63	44	25	12	11	46
54	60	1	86	64	9	78	86	2	38	85	10	93	45	7	34
71	23	75	64	25	97	65	30	52	29	75	77	93	38	23	47
98	77	86	63	86	16	1	40	55	38	2	0	35	61	35	58

```
$ ./maxsumrec < input.txt
Running maxsumrec
Max sum is 1146
$
```

Μπορείτε να δοκιμάσετε το πρόγραμμά σας και με άλλους πίνακες, που γεννώνται τυχαία από το πρόγραμμα “randmatr_<arch>”, όπου το <arch> είναι linux, windows.exe ή macosx, ανάλογα με το σύστημα που σας ενδιαφέρει. Τα εκτελέσιμα αυτού του προγράμματος για τις προηγούμενες αρχιτεκτονικές μπορείτε να τα βρείτε στο <http://www.di.uoa.gr/~ip/hwfiles/maxsum>. Το πρόγραμμα “randmatr_<arch>” μπορεί να κληθεί είτε χωρίς ορίσματα, είτε από ένα έως τέσσερα ορίσματα, και παράγει στην έξοδό του ένα τυχαίο πίνακα στη μορφή που τον περιμένει το πρόγραμμά σας (πρώτα οι διαστάσεις του πίνακα και μετά, κατά γραμμές, τα στοιχεία του). Οι διαφορετικές εκδοχές χρήσης του προγράμματος περιγράφονται στη συνέχεια:

- **randmatr_<arch>**: Δημιουργεί ένα τυχαίο πίνακα με 10 γραμμές και 10 στήλες, με στοιχεία που έχουν τιμές από 0 έως 99 και με φύτρο της γεννήτριας τυχαίων αριθμών τον τρέχοντα χρόνο.
- **randmatr_<arch> <N>**: Δημιουργεί ένα τυχαίο πίνακα με <N> γραμμές και 10 στήλες, με στοιχεία που έχουν τιμές από 0 έως 99 και με φύτρο της γεννήτριας τυχαίων αριθμών τον τρέχοντα χρόνο.
- **randmatr_<arch> <N> <M>**: Δημιουργεί ένα τυχαίο πίνακα με <N> γραμμές και <M> στήλες, με στοιχεία που έχουν τιμές από 0 έως 99 και με φύτρο της γεννήτριας τυχαίων αριθμών τον τρέχοντα χρόνο.
- **randmatr_<arch> <N> <M> <Maxvalue>**: Δημιουργεί ένα τυχαίο πίνακα με <N> γραμμές και <M> στήλες, με στοιχεία που έχουν τιμές από 0 έως <Maxvalue>-1 και με φύτρο της γεννήτριας τυχαίων αριθμών τον τρέχοντα χρόνο.
- **randmatr_<arch> <N> <M> <Maxvalue> <Seed>**: Δημιουργεί ένα τυχαίο πίνακα με <N> γραμμές και <M> στήλες, με στοιχεία που έχουν τιμές από 0 έως <Maxvalue>-1 και με φύτρο της γεννήτριας τυχαίων αριθμών το <Seed>.

Κάποιες επιπλέον εκτελέσεις³ του προγράμματος “maxsum” με τυχαίες εισόδους που παράγονται από το πρόγραμμα “randmatr_linux”, φαίνονται στη συνέχεια.

```
$ ./randmatr_linux 12 16 200 134 | ./maxsumrec
Running maxsumrec
Max sum is 1896
$
$ ./randmatr_linux 15 20 150 4 | ./maxsumrec
Running maxsumrec
Max sum is 1821
$
$ ./randmatr_linux 17 13 400 2294 | /usr/bin/time ./maxsumrec
```

³σε μηχανήμα Linux του εργαστηρίου του Τμήματος

```

Running maxsumrec
Max sum is 5208
4.23user 0.00system 0:04.26elapsed 99%CPU .....
$
$ ./randmatr_linux 18 18 1000 123 | /usr/bin/time ./maxsumrec
Running maxsumrec
Max sum is 14892
17.79user 0.00system 0:17.80elapsed 99%CPU .....
$
$ ./randmatr_linux 20 22 10 1 | /usr/bin/time ./maxsumrec
Running maxsumrec
Max sum is 163
196.46user 0.00system 3:16.47elapsed 99%CPU .....
$

```

Αναδρομική μέθοδος με απομνημόνευση (25%)

Παρατηρείτε ότι όσο μεγαλώνει ο πίνακας, τόσο πιο πολύ αργεί η εκτέλεση του προγράμματός σας που εφαρμόζει κατά γράμμα τον αναδρομικό τύπο που δόθηκε; Και μάλιστα, η αύξηση του χρόνου εκτέλεσης γίνεται με μεγάλο ρυθμό, που ονομάζεται εκθετικός, και είναι τέτοιος που από κάποιο μέγεθος πίνακα και πάνω, το πρόγραμμά σας πρακτικά δεν δίνει καθόλου αποτελέσματα. Όπως ίσως μπορείτε να καταλάβετε, αυτό οφείλεται στο ότι στην αναδρομική συνάρτηση που έχετε γράψει, καλείται ο εαυτός της τρεις φορές. Πώς θα μπορούσαμε να το διορθώσουμε αυτό; Αρκεί να χρησιμοποιήσουμε ένα δεύτερο διδιάστατο πίνακα $n \times m$, στον οποίο να αποθηκεύεται κάθε S_{ij} την πρώτη φορά που υπολογίζεται και όταν χρειάζεται πάλι, να μην επαναυπολογίζεται, αλλά να λαμβάνεται η τιμή του από τον πίνακα. Η λογική της βελτιωμένης μεθόδου εξακολουθεί να είναι αναδρομική, βασισμένη στη μαθηματική σχέση που δόθηκε, απλώς κάθε S_{ij} υπολογίζεται ακριβώς μία φορά και φυλάσσεται στον πίνακα για μελλοντική χρήση. Υλοποιήστε τη συνάρτηση `solve()` και με βάση αυτήν την προσέγγιση (αρχείο `maxsummem.c`).

Παραδείγματα εκτέλεσης για τη νέα υλοποίηση είναι τα εξής:

```

$ ./maxsummem
6 5
7 4 5 2 7
6 9 1 6 3
9 2 7 9 8
8 8 1 7 2
2 3 1 8 4
4 6 5 4 6
Running maxsummem
Max sum is 44
$
$ ./maxsummem
6 5
7 4 5 2 7
6 9 1 6 3
9 2 7 9 8
8 8 1 7 9
2 3 1 8 8

```

```

4 6 5 4 7
Running maxsummem
Max sum is 49
$
$ ./maxsummem < input.txt
Running maxsummem
Max sum is 1146
$
$ ./randmatr_linux 20 22 10 1 | ./maxsummem
Running maxsummem
Max sum is 163
$
$ ./randmatr_linux 100 100 100 999 | ./maxsummem
Running maxsummem
Max sum is 8306
$
$ ./randmatr_linux 1000 1000 100 999 | ./maxsummem
Running maxsummem
Max sum is 81700
$
$ ./randmatr_linux 10000 10000 100 999 | ./maxsummem
Running maxsummem
Max sum is 815519
$

```

Παρατηρήστε ότι πλέον το πρόγραμμά σας δουλεύει πολύ γρήγορα, ακόμα και για πολύ μεγάλους πίνακες εισόδου. Εύλογο δεν είναι;

Επαναληπτική μέθοδος με δυναμικό προγραμματισμό (25%)

Ένας εναλλακτικός τρόπος για να αντιμετωπίσετε το πρόβλημα είναι με τη βοήθεια μίας μη-αναδρομικής συνάρτησης, η οποία απλώς θα συμπληρώνει τον πίνακα των μερικών αθροισμάτων με μία επαναληπτική διαδικασία. Η τεχνική αυτή ονομάζεται *δυναμικός προγραμματισμός*. Σκεφτείτε, όμως, με ποια σειρά θα πρέπει να υπολογίζονται τα στοιχεία του πίνακα και αν αυτή πρέπει να είναι ίδια ή διαφορετική από τη σειρά με την οποία υπολογίζονται τα ενδιάμεσα αθροίσματα στην προηγούμενη μέθοδο. Κάνετε και μία τρίτη υλοποίηση της συνάρτησης `solve()`, βασισμένη σε αυτή τη λογική (αρχείο `maxsumdp.c`). Τα αποτελέσματα που θα έχετε δεν θα πρέπει να διαφέρουν από αυτά της προηγούμενης μεθόδου. Για παράδειγμα:

```

$ ./maxsumdp < input.txt
Running maxsumdp
Max sum is 1146
$
$ ./randmatr_linux 10000 10000 100 999 | ./maxsumdp
Running maxsumdp
Max sum is 815519
$

```

Υπολογισμός των βέλτιστων μονοπατιών (25%)

Να επεκτείνετε την αναδρομική υλοποίηση με απομνημόνευση και την επαναληπτική υλοποίηση με δυναμικό προγραμματισμό έτσι ώστε να υπολογίζονται και τα βέλτιστα μονοπάτια (εκτός από τα βέλτιστα αθροίσματα). Αν υπάρχουν περισσότερα από ένα μονοπάτια με το ίδιο μέγιστο άθροισμα, το πρόγραμμά σας αρκεί να εκτυπώνει ένα μόνο από αυτά. Το αν θα εκτυπώνονται ή όχι τα μονοπάτια να εξαρτάται από το αν έχει ορισθεί η συμβολική σταθερά `PATH`. Τον κώδικα εκτύπωσης των μονοπατιών στη συνάρτηση θα πρέπει να τον περικλείσετε μέσα στις οδηγίες προς τον προεπεξεργαστή `"#ifdef PATH"` και `"#endif"` (δείτε τις σελίδες 158-159 των σημειώσεων/διαφανειών του μαθήματος). Αν θέλετε να εκτυπώνονται και τα βέλτιστα μονοπάτια, θα πρέπει είτε να έχετε κάνει `#define` την `PATH`, είτε να μεταγλωττίσετε το πρόγραμμά σας με την εντολή `"gcc -DPATH ..."` ώστε να μεταγλωττιστεί και ο κώδικας της εκτύπωσης των μονοπατιών. Παραδείγματα εκτέλεσης με υπολογισμό μονοπατιών είναι τα εξής:

```
$ gcc -DPATH -o maxsummempath maxsum.c maxsummemp.c
$ gcc -DPATH -o maxsumdppath maxsum.c maxsumdp.c
$
$ ./maxsummempath
6 5
7 4 5 2 7
6 9 1 6 3
9 2 7 9 8
8 8 1 7 2
2 3 1 8 4
4 6 5 4 6
Running maxsummempath
Max sum is 44
7 -> 9 -> 7 -> 7 -> 8 -> 6
$
$ ./maxsummempath
6 5
7 4 5 2 7
6 9 1 6 3
9 2 7 9 8
8 8 1 7 9
2 3 1 8 8
4 6 5 4 7
Running maxsummempath
Max sum is 49
7 -> 9 -> 9 -> 9 -> 8 -> 7
$
$ ./maxsummempath < input.txt
Running maxsummempath
Max sum is 1146
78 -> 96 -> 72 -> 72 -> 71 -> 93 -> 83 -> 92 -> 82 -> 96 -> 64 -> 86 -> 75 -> 86
$
$ ./randmatr_linux 20 22 10 1 | ./maxsummempath
Running maxsummempath
Max sum is 163
7 -> 9 -> 8 -> 9 -> 9 -> 6 -> 7 -> 8 -> 7 -> 9 -> 9 -> 9 -> 8 -> 9 -> 9 -> 9 ->
```

```

9 -> 6 -> 7 -> 9
$
$ ./randmatr_linux 100 100 100 999 | ./maxsummempath
Running maxsummempath
Max sum is 8306
90 -> 85 -> 97 -> 64 -> 72 -> 80 -> ..... -> 89 -> 91 -> 89 -> 92 -> 74 -> 71
$
$ ./randmatr_linux 1000 1000 100 999 | ./maxsummempath
Running maxsummempath
Max sum is 81700
81 -> 95 -> 87 -> 59 -> 91 -> 98 -> ..... -> 91 -> 96 -> 90 -> 82 -> 96 -> 81
$
$ ./randmatr_linux 10000 10000 100 999 | ./maxsummempath
Running maxsummempath
Max sum is 815519
82 -> 86 -> 75 -> 89 -> 88 -> 91 -> ..... -> 93 -> 99 -> 83 -> 99 -> 94 -> 52
$
$ ./maxsumdppath < input.txt
Running maxsumdp
Max sum is 1146
78 -> 96 -> 72 -> 72 -> 71 -> 93 -> 83 -> 92 -> 82 -> 96 -> 64 -> 86 -> 75 -> 86
$
$ ./randmatr_linux 10000 10000 100 999 | ./maxsumdppath
Running maxsumdp
Max sum is 815519
82 -> 86 -> 75 -> 89 -> 88 -> 91 -> ..... -> 93 -> 99 -> 83 -> 99 -> 94 -> 52
$

```

Παραδοτέο

Θα πρέπει να δομήσετε το πρόγραμμά σας σε ένα σύνολο από **πηγαία αρχεία C** (με κατάληξη `.c`), ένα με τη συνάρτηση `main()` και από ένα για κάθε μία από τρεις μεθόδους που ζητείται να υλοποιήσετε, και **τουλάχιστον ένα αρχείο επικεφαλίδας** (με κατάληξη `.h`). Τυχόν βοηθητικές συναρτήσεις που θα χρειαστεί να υλοποιήσετε μπορούν να περιληφθούν είτε σε κάποια από τα προηγούμενα πηγαιά αρχεία, είτε σε κάποιο άλλο.

Για να παραδώσετε το σύνολο των αρχείων που θα έχετε δημιουργήσει για την εργασία αυτή, ακολουθήστε την εξής διαδικασία. Τοποθετήστε όλα τα αρχεία⁴ μέσα σ' ένα κατάλογο που θα δημιουργήσετε σε κάποιο σύστημα Linux, έστω με όνομα `maxsum`. Χρησιμοποιώντας την εντολή `zip` ως εξής

```
zip -r maxsum.zip maxsum
```

δημιουργείτε ένα συμπιεσμένο (σε μορφή `zip`) αρχείο, με όνομα `maxsum.zip`, στο οποίο περιέχεται ο κατάλογος `maxsum` μαζί με όλα τα περιεχόμενά του.⁵ Το αρχείο αυτό είναι που θα πρέπει να υποβάλετε μέσω του `eclass`.⁶

⁴πηγαία `.c` και επικεφαλίδας `.h`, **όχι εκτελέσιμα ή αντικειμενικά**

⁵Αρχεία `zip` μπορείτε να δημιουργήσετε και στα Windows, με διάφορα προγράμματα, όπως το 7-zip ή το WinZip.

⁶Μην υποβάλετε ασυμπίεστα αρχεία ή αρχεία που είναι συμπιεσμένα σε άλλη μορφή εκτός από `zip` (π.χ. `rar`, `7z`, `tar`, `gz`, κλπ.), γιατί δεν θα γίνουν δεκτά για αξιολόγηση.