a)

Για την υλοποιηση του Β ερωτηματος διαβαζουμε μια μια τις γραμμες απο το input αρχειο χρησομοποιωντας την BufferedReader στα προτυπα της εργασιας 1. Στη συνεχεια απο τις δυο πρωτες γραμμες εξαγουμε τον αριθμο των επεξεργασστων και τον αριθμο των διεργασιως , τις τιμες αυτες τις αποθηκευουμε σε ξεχωριστες μεταβλητες καθως θα μας χρειαστουν στη πορεια. Απο εδω και περα εχουμε εναν μετρητη που χρησιμευει στο να μετρησει ποσες διεργασιες εχει πραγματικα το αρχειο. Αν ο αριθμος του μετρητη οταν κλεισει το αρχειο δεν ειναι ιδιος με τον αριθμο των διεργασιων που ειχαμε εξαγει στην αρχη τυπωνεται μηνυμα σφαλματος. Αναλογα με το ποσους επεξεργαστες εχουμε διαβασει στο input οτι θελουμε , τοσους επεξεργαστες θα δημιουργησουμε και θα εισαγουμε απευθειας στην ουρα προτεραιοτητας.Τα ζευγη τιμων id – χρονου της καθε διεργασιας που διαβαζουμε αποθηκευονται σε ενα αντικειμενο τυπου Task και ενα ενα εισαγονται στη λιστα(generic παιρνει αντικειμενα τυπου Task) του επεξεργαστη που μας δινει η ουρα μετα απο κληση της getMax(). Τον επεξεργαστη που δινει η getmax() τον ξαναεισαγουμε στη λιστα με insert. Στο τελος με κληση της getmax τυπωνουμε ουσιαστικα με τη σειρα που θελουμε τους επεξεργαστες και παιρνωντας τον συνολικο χρονο του τελευταιου επεξεργαστη που βγαινει απο τη λιστα εχουμε το makespan. Για λογους ευκολιας για το Δ ερωτημα εχουμε φτιαξει την μεθοδο greedy στο τελος να επιστρεφει το makespan (πραγμα αχρειαστο στο Β ερωτημα).

b)

Ταξινομησαμε τις διεργασιες μας με χρηση HeapSort. Η υλοποιηση εγινε με χρηση πίνακα. Το προγραμμα μας δεν εξαρταται απο το Β μερος της εργασιας. Διαβαζουμε απο την αρχη μια μια τις γραμμες του input και εισαγουμε στον πινακα μια μια τις διεργασιες. Στο τελος περναμε ουσιαστικα απο τον αλγοριθμο του Β ερωτηματος τον πινακα με τις ταξινομημενες διεργασιες. Οπως και στο Β ερωτημα ετσι και εδω ια λογους ευκολιας για το Δ ερωτημα εχουμε φτιαξει την μεθοδο στο τελος να επιστρεφει το makespan (πραγμα αχρειαστο στο Γ ερωτημα).

c)

Στο μέρος Δ και στην κλάση Comparisons φτιάξαμε ένα πρόγραμμα που δημουργεί 30 συνολικά αρχεία txt με επεξεργαστές και διεργασίες σύμφωνα με τα ζητούμενα, ο χρόνος για τις διεεργασίες έγινε με τυχαίο τρόπο από το 1-200 (int task_time = rand.nextInt(200);). Κάθε φορά που τρέχει η Comparisons δημιουργεί καινούρια 30 αρχεία. Σε όλα τα πειράματα αποδείξαμε ότι ο Algorithm 2 είναι σημαντικά γρηγορότερος, παρακατω φαινονται ενδεικτικα καποια αποτελεσματα.

```
<terminated> Comparisons [Java Application] C:\Program Files\Java\jre1.8.0_144\bin\javaw.exe (28 Δεκ 2021, 5:52:31 μ.μ.)
Number of Tasks = 100
Average Makespan using Algorithm 1 = 105.25
Average Makespan using Algorithm 2 = 98.04
Number of Tasks = 225
Average Makespan using Algorithm 1 = 116.83111111111111
Average Makespan using Algorithm 2 = 110.02222222222223
Number of Tasks = 400
Average Makespan using Algorithm 1 = 117.6875
Average Makespan using Algorithm 2 = 111.49
```

```
<terminated> Comparisons [Java Application] C:\Program Files\Java\jre1.8.0_144
Number of Tasks = 100
Average Makespan using Algorithm 1 = 106.78
Average Makespan using Algorithm 2 = 98.32
Number of Tasks = 225
Average Makespan using Algorithm 1 = 117.05333333333333
Average Makespan using Algorithm 2 = 109.14222222222222
Number of Tasks = 400
Average Makespan using Algorithm 1 = 118.3325
Average Makespan using Algorithm 2 = 111.355
```

```
<terminated> Comparisons [Java Application] C:\Program Files\Java\jre1.8.0_144\bin\javaw.exe (28 Δεκ 2021, 5:53:33 μ.μ.)
Number of Tasks = 100
Average Makespan using Algorithm 1 = 109.48
Average Makespan using Algorithm 2 = 102.49
Number of Tasks = 225
Average Makespan using Algorithm 1 = 119.11111111111111
Average Makespan using Algorithm 2 = 112.62222222222222
Number of Tasks = 400
Average Makespan using Algorithm 1 = 119.1925
Average Makespan using Algorithm 2 = 113.4075
```

```
<terminated> Comparisons [Java Application] C:\Program Files\Java\jre1.8.0_144\bin\java
Number of Tasks = 100
Average Makespan using Algorithm 1 = 108.23
Average Makespan using Algorithm 2 = 101.36
Number of Tasks = 225
Average Makespan using Algorithm 1 = 116.26222222222222
Average Makespan using Algorithm 2 = 109.50222222222222
Number of Tasks = 400
Average Makespan using Algorithm 1 = 117.425
Average Makespan using Algorithm 2 = 111.395
```

Τα αρχεια που δημιουργηθηκαν και εδωσαν τα αποτελεσματα της τελευταιας εικονας βρισκονται μεσα στον φακελο data που παραδωθηκε μαζι με τον κωδικα της εργασιας. Καθε φορα που τρεχει το Comparisons.java δημιουργουνται 30 αρχεια με τυχαιες τιμες τα οποια μπορειτε να τα βρειτε στο φακελο του project μεσα στο οποιο βρισκεται και ο φακελος src με τον κωδικα. Άρα, μπορεί να διεξαχθεί ασφαλές συμπέρασμα ότι ο αλγόριθμος 2 είναι αρκετά αποτελεσματικότερος σε ότι αφορά στον χρόνο διεργασίας. Μια επιπλέον παρατήρηση είναι ότι όσο μεγαλώνει το N δείχνει ότι ο χρόνος μεταξύ των αλγορίθμων οδηγείται σε σταθεροποίηση κοντά στα 6s. Τελικό συμπέρασμα της μελέτης είναι ότι ο ταξινομημένος πίνακας έχει σαφές πλεονέκτημα έναντι του μη ταξινομημένου πίνακα σε συνάρτηση του χρόνου.

d

Για να τρεξουν τα προγραμματα χρησιμοποιουμε arguments για να λαβουμε αρχεια input. Τα αρχεια αυτα πρεπει να βρισκονται μεσα στον φακελο src μαζι με τον κωδικα. Ενδεικτικα μετα την μεταγλώττιση μπορει να τρεξει με εντολη τυπου java Greedy input_file.txt ή όπως το τρέχαμε εμείς μεσω Eclipse με τον τροπο που φαινεται παρακάτω: