Χρήστος Ξυδέας (Αρ. Μητρώου: 3160124)
Δικαία Σωτηροπούλου (Αρ. Μητρώου: 3160172)

Μέρος Α' και Γ'

Στο Α' μέρος δημιουργούμε μια διπλά συνδεδεμένη λίστα τύπου CharDoubleEndedQueue, η οποία περιέχει αντικείμενα τύπου Node με τα ονόματα *head* (πρώτος κόμβος λίστας), *tail* (τελευταίος κόμβος λίστας), *next* (διεύθυνση επόμενου κόμβου) και *previous* (διεύθυνση προηγούμενου κόμβου). Κάθε *Node* περιέχει τα δεδομένα τύπου *char* (*info*).
Οι λειτουργίες μεθόδων που μπορούν να υλοποιηθούν στη λίστα *CharDoubleEndedQueue* είναι οι ακόλουθες:

isEmpty() : ελέγχει αν η λίστα είναι άδεια επιστρέφοντας μια τιμή τύπου boolean ανάλογα με το αν το μέγεθος της λίστας είναι 0 (true) ή όχι (false),
addFirst(char item) : προσθέτει ένα νέο στοιχείο τύπου *char* (*item*) δημιουργώντας ένα νέο κόμβο (*newNode*) στην αρχή της λίστας,
removeFirst() : αφαιρεί τον πρώτο κόμβο της λίστας, χρησιμοποιώντας μια βοηθητική μεταβλητή (*temp*). Στην περίπτωση που η λίστα είναι άδεια το πρόγραμμα πετάει εξαίρεση τύπου *NoSuchElementException*,
addLast(char item) : προσθέτει ένα νέο στοιχείο τύπου char (*item*) δημιουργώντας ένα νέο κόμβο (*newNode*) στο τέλος της λίστας,
removeLast() : αφαιρεί τον τελευταίο κόμβο της λίστας, χρησιμοποιώντας μια βοηθητική μεταβλητή (*temp*). Στην περίπτωση που η λίστα είναι άδεια το πρόγραμμα πετάει εξαίρεση τύπου *NoSuchElementException*,
getFirst() : επιστρέφει τα δεδομένα του πρώτου κόμβου της λίστας, χωρίς να αφαιρεί τον κόμβο. Στην περίπτωση που η λίστα είναι άδεια το πρόγραμμα πετάει εξαίρεση τύπου *NoSuchElementException*,
getLast() : επιστρέφει τα δεδομένα του τελευταίου κόμβου της λίστας, χωρίς να αφαιρεί τον κόμβο. Στην περίπτωση που η λίστα είναι άδεια το πρόγραμμα πετάει εξαίρεση τύπου *NoSuchElementException*,
printQueue(PrintStream stream) : εκτυπώνει την λίστα από την αρχή προς το τέλος ,εκτός αν είναι άδεια που τότε τυπώνει ανάλογο μήνυμα,
size() : επιστρέφει το μέγεθος της λίστας.

Στο Γ' μέρος δημιουργούμε μια απλά συνδεδεμένη λίστα τύπου *CharQueue*, η οποία περιέχει αντικείμενα τύπου *Node* με τα ονόματα *head* (πρώτος κόμβος λίστας), *tail* (τελευταίος κόμβος λίστας), *next* (διεύθυνση επόμενου κόμβου). Κάθε *Node* περιέχει τα δεδομένα τύπου char (*info*).Οι λειτουργίες μεθόδων που μπορούν να υλοποιηθούν στη λίστα *CharQueue* είναι οι ακόλουθες:

isEmpty() : ελέγχει αν η λίστα είναι άδεια επιστρέφοντας μια τιμη τύπου boolean ανάλογα με το αν το μέγεθος της λίστας είναι 0 (true) ή όχι (false),
put (char item) : προσθέτει ένα νέο στοιχείο τύπου char (*item*) δημιουργώντας ένα νέο κόμβο (*newNode*) στο τέλος της λίστας,

Χρήστος Ξυδέας (Αρ. Μητρώου: 3160124)
Δικαία Σωτηροπούλου (Αρ. Μητρώου: 3160172)

get() : αφαιρεί το παλαιότερο στοιχείο της λίστας το οποίο είναι το πρώτο στοιχείο ,και το επιστρέφει στο πρόγραμμα. Στην περίπτωση που η λίστα είναι άδεια το πρόγραμμα πετάει εξαίρεση τύπου *NoSuchElementException*,
peek() : επιστρέφει το παλαιότερο στοιχείο της λίστας χωρίς να το διαγράφει. Στην περίπτωση που η λίστα είναι άδεια το πρόγραμμα πετάει εξαίρεση τύπου *NoSuchElementException*,
printQueue(PrintStream stream) : εκτυπώνει την λίστα από την αρχή προς το τέλος ,εκτός αν είναι άδεια που τότε τυπώνει ανάλογο μήνυμα,
size() : επιστρέφει το μέγεθος της λίστας.

Μέρος Β'

Στο δεύτερο μέρος της εργασίας κατασκευάσαμε, με την βοήθεια του πρώτου μέρους ,ένα πρόγραμμα το οποίο δέχεται μια αλυσίδα DNA και ελέγχει αν ανήκει στην κατηγορία Watson-Crick complemented palindrome. Ο έλεγχος αυτός γίνεται ως εξής:

Αρχικά ζητείται από τον χρήστη να πληκτρολογήσει μια αλυσίδα DNA. Το πρόγραμμα αποθηκεύει την αλυσίδα σε μια μεταβλητή τύπου *String* με όνομα DNA και στη συνέχεια με την χρήση της μεθόδου checkDNA (char character) ελέγχει αν έχει δοθεί σωστή ακολουθία DNA (αν δεν υπάρχουν πεζοί χαρακτήρες, αν υπάρχουν μόνο οι χαρακτήρες A, T, G, C, αν δεν περιέχονται κενά) και συνεχίζει αλλιώς το πρόγραμμα τερματίζει. Έπειτα, δημιουργεί μια δίπλα συνδεδεμένη λίστα τύπου *CharDoubleEndedQueueImpl* με όνομα *listDNA* (από το Α' μέρος) και προσθέτει στο τέλος της λίστας το συμπλήρωμα του κάθε χαρακτήρα (char) από την δοσμένη αλυσίδα του χρήστη, με την βοήθεια της μεθόδου *Supplement (char character)* που δημιουργήσαμε. Έχοντας αντιστρέψει τη *listDNA*, σε μια καινούργια μεταβλητή τύπου *StringBuilder* με όνομα *WatsonCrickDNA* μεταφέρεται η αντεστραμμένη αυτή λίστα. Τέλος συγκρίνεται η αρχική αλυσίδα DNA (*DNA*) με την νέα αυτή μεταβλητή(*WatsonCrickDNA*). Αν είναι ίδιες σημαίνει ότι η αρχική αλυσίδα DNA, είναι τύπου Watson-Crick complemented palindrome τυπώνοντας αντίστοιχο μήνυμα.

Μέρος Δ'

Στο τέταρτο μέρος της εργασίας κατασκευάσαμε με την βοήθεια του Α' και Γ' μέρους που υλοποιήθηκαν, το interface το οποίο επιστρέφει το μικρότερο στοιχείο μιας λίστας, με τη χρήση της μεθόδου *min()*.Οι μέθοδοι που χρησιμοποιούνται είναι οι εξής:

Χρήστος Ξυδέας (Αρ. Μητρώου: 3160124)
Δικαία Σωτηροπούλου (Αρ. Μητρώου: 3160172)

Min(): επιστρέφει το μικρότερο στοιχείο της λίστας δηλαδή το πρώτο στοιχείο της λίστας τύπου *charDoubleEndedQueue* με όνομα *doublyLinkedList*,
Get(): αφαιρεί το παλαιότερο στοιχείο της λίστας δηλαδή το πρώτο στοιχείο της *doublyLinkedList*. Στην περίπτωση που η λίστα είναι άδεια το πρόγραμμα πετάει εξαίρεση τύπου *NoSuchElementException*,
Put (char item): προσθέτει ένα νέο στοιχείο (*item*) στην λίστα τύπου *CharQueueImpl* με όνομα *queue*. Αν η διπλά συνδεδεμένη λίστα είναι άδεια ή αν το *item* είναι μεγαλύτερο από το τελευταίο στοιχείο της *doublyLinkedList*, τότε το προσθέτει στο τέλος της τελευταίας , αλλιώς αφαιρεί τα μεγαλύτερα στοιχεία μέχρις ότου το τελευταίο στοιχείο να είναι μικρότερο από το item.

Επίσης, δημιουργήσαμε ένα επιπλέον κομμάτι κώδικα με όνομα *Test_Main_With_Min* το οποίο χρησιμεύει στη λειτουργία του προγράμματος του Δ' μέρους.