PART A

isEmpty: this function checks if the queue is empty and returns true if it is, or false if it's not.

addFirst: this function adds a character at the front side of the queue

removeFirst: this function removes the front element of the queue and then it returns the character. Also if the queue is empty it prints an error message

addLast: this function adds a character at the rear side of the queue

removeLast: this function removes the rear element of the queue and then it returns the character. Also if the queue is empty it prints an error message

getFirst: this function returns the first item of the queue and if the queue is empty

it prints an error message

getLast: this function returns the last item of the queue and if the queue is empty

it prints an error message

printQueue: this function prints all the character of the queue starting from front

size: this function returns the size of the queue

PART B

At first we made a new class called DNAPalindrome and then created a new queue called A, then with the help of the user we made a sequence with the 4 available letters(with the number 0 the program stopped reading characters) and we added the string to the queue A. After that we made a string variable to store our sequence called initial, then with the help of a loop we changed the string depending on the letter(A->T, C->G,..). Now to see if this sequence was WATSON-CRICK we needed to reverse the string and in order to do that we made a while that stops at the half size of the string, and we were changing each character with the other character(1st with last,2nd with second last and go on), also we used a variable called tmp to help us store the character before it was changed.

In the end when we finished reversing the string we stored it to a string variable called finale and then we compared it with initial, if it was true the the sequence was watson or else it wasn't.

PART C

isEmpty: this function checks if the queue is empty and returns true if it is, or false if it’s not

put: this function inserts a character at the start of the queue

get: this function returns the oldest item of the queue and if the queue is empty it prints an error message

peek: this function returns and removes the oldest item of the queue and if the queue is empty it prints an error message

printQueue: this function prints all the characters of the queue starting from front

size: this function returns the size of the queue

PART D

In this part of the assignment we created 2 Queues.

The F Queue(FIFO): used for storage of chars

The D Queue(DoubleEnded): used to speed the process of finding the minimum value

Put: This function adds the selected character to the F queue and then, in order to make this process faster:

- In each of the List Nodes we added a counter variable, which counted, in this case, how many times we have seen this character being added to the queue.
- Afterwards, if this character is not in the D queue, we add it in the beginning
- If it is in the queue already, we just do node counter++; So we know how many times each element has been presented.
- For example if I give input (A-A-A-B) it will be registered something like (3-A, 1-B) and the queue now only has 2 elements instead of 4.
- Moreover, the put function tests whether the new character is a new minimum, by comparing it to the old one

Get: This function does the following:

- F.get();
- Then it checks where is that particular char in the D queue and once it finds it, it makes the node counter--;
- We also added a Boolean ignore “ignore” variable which is assigned a default value of false. When the counter==0 ignore is set to true and when we scan the queue for a new min these nodes are ignored in the process, effectively cutting down its size.
- However: if the start or the end queue is to be ignored, we delete it instead.
- After removing the element, we find the new min in the queue using a while, but the queue is significantly smaller and more efficient than a normal queue through this process.

Min: This function returns the variable “min” defined in the program.

The rest of the functions work as described in previous parts (printQueue, peek, size).