How to use the program:

1. Install Eclipse (and JDK)

(<https://www.ntu.edu.sg/home/ehchua/programming/howto/EclipseJava_HowTo.html> - might be a good start).

2. Download the program from here: <https://github.com/MihaiAC/Edinburgh_UG-iGem2017-OligoDesigner> .

3. Open Eclipse and import the program (project) as seen here: <http://help.eclipse.org/kepler/index.jsp?topic=%2Forg.eclipse.platform.doc.user%2Ftasks%2Ftasks-importproject.htm> .

4. Open Main.java in Eclipse and locate the run configurations menu. <https://www.tutorialspoint.com/eclipse/eclipse_run_configuration.htm>

5. Under the “Arguments” tab in Run Configurations, paste your input sequence, between “ ”, followed in order by: the minimum length of a repeat, the minimum length of a part, the maximum length of a part and another integer.

Repeats are generally to be contained in separate parts. This last integer relaxes this constraint: a part can contain one repeated sequence two times, provided that its length is less than or equal this integer.

6. Click Run and one of three things happen:

- the program doesn’t find any optimal split;

- the program finds a split which satisfies all the conditions;

- the program runs out of time (3 minutes is the max runtime limit);

In any of the three cases the output of the program can be found in the file “output.txt”, which will be in the same folder as the project. This file will contain the detected repeats and where each occurs. It will also contain the parts, if a split was found.

NOTE:

The program was tested only on sequences of max length = 400 and only on our sequences (so there may be bugs).

What the program does (specifications):

Oligo Designer Program

Input: See Part 5 above.

Output: Oligos in 5’ to 3’ direction (a.k.a. sequence of the parts, upper strand read from left to right, bottom strand read right to left)

1. The pasted sequence is only of the upper strand.
2. The program should identify repeats in the sequence that are larger than 15 base pairs (the number should be able to be modified).
3. There should be a fusion site between repeats.

A fusion site

* is 4 letters long
* Has at least one G or C
* Has at least one A or T
* It isn’t palindromic with other fusion sites (e.g. AGCT is palindromic, its complementary fusion site (its bottom strand is TCGA ->, hence read backwards as AGCT)
* It is sufficiently different from other fusion sites, meaning
  + - * + There can only be 2 letters in the same position at max (e.g. AGTT and AGAT are unacceptable together – there are 3 letters in the same position)

*(when considering fusion sites for ‘difference’ and ‘palindrome’, the bottom strand fusion site is read from right to left)*

*e.g., AGTT and ATCT are also unacceptable; reason: difference, as the reverse complement of ATCT is AGAT (bottom strand of ATCT is TAGA, read backwards: AGAT). ATCT and AGAT aren’t sufficiently different.*

1. If a repeat of relevant has two instances, one in the forward direction (left to right) and the other in the backward direction (right to left), they should be in different parts too.
2. The fusion sites separate the input sequence into parts. The parts should be at the length of 45-200 letters.
3. The program should prefer solutions that require separating the sequence into least number of parts.
4. There should be an annotation of where the sequence is separated into parts.

How does it work? (overview)

1. All repeats are detected efficiently as presented here: <https://arxiv.org/pdf/1304.0528.pdf> .

This method makes use of the suffix array of the sequence, which is constructed in the way seen here: <http://algo2.iti.kit.edu/documents/jacm05-revised.pdf> .

2. The program identifies all possible fusion sites in the sequence (and their positions) and constructs a constraint graph. Two fusion sites are connected by an edge of the graph if they cannot be selected together (due to the constraints).

3. Here, the problem has become a constraint satisfaction problem (CSP - <https://www.inf.ed.ac.uk/teaching/courses/inf2d/slides/08_CSP_wbg.pdf> ). Every possible (and valid) combination of fusion sites is tested. The number of fusion sites in a combination increases from 1 to floor(length of the sequence)/(minimum length of a part). If a combination of fusion sites satisfies all the constraints, then it is immediately returned and the program stops after it outputs the parts.

Future work/possible improvements:

* CSPs are always greatly improved by the right heuristic/combination of heuristics; different heuristics should be tried here in order to improve the runtime of the program.