SOP Project – Documentation

1) Base features

a) *Insert* function: The *insert* function adds new triplets to the list.
After verifying if the input is of the correct type and a new queue is created if necessary, the parameters are added via the use of a temp node. If the queue was empty, front and rear are assigned. Otherwise, the new node is added at the end and the rear pushed back.

b) *Match* function: The *match* function finds the x^{th} (x is called *result*) triplet in the list that corresponds to the searched parameter(s).
To find matches, the function first asses how many parameters (1 to 3) are to be compared. It then uses a temporary node to go through the list and compare parameters to list triplets. The function uses a countdown as high as the "*result*" variable that gets lowered with each match found until it gets to 0 and the corresponding match is printed.
If no corresponding match is found, the function asses if this issue comes from an actual lack of match or from a lack of enough results

2) Additional features

a) *Erase* function: By using a method similar to the *match* function, the desired triplet is found in the list and then erased.
Unlike *match*, the triplet to be erased has to be given in full. After making sure all parameters are correct, the function finds the desired triplet by comparing the parameters using a node. Once the triplet is found, the node jumps over the triplet and the memory is freed.
If the triplet to erase is at the end of the queue, the rear is moved forward after erasing the triplet

b) *Print_list* function: The function uses a node to go through the queue and print each triplet in it.

3) Architecture and decisions

Our project uses Linked lists to hold the data. We also implemented a simple *init()* function to create a new queue easily as well as a *library* to declare all of the variables and functions used.

Each file uses *if* statements to assess the various possible cases according to the function. They also use temporary nodes to go through the list to find triplets. Each function is declared and defined in its own file and imported in other files through the *library*.
According to the results of the Benchmark, our implementation is efficient with *insert* taking about 1800ms and *match* only 1ms

4) Testing

A *test.c* file with a *main(int argc, char *argv[])* function has been built to test the different features of the project. The function creates a new queue, asks for personalised input and uses the *insert* function to fill the queue with triplets (from the input as well as pre-added). It then uses the *match* function to find various triplets and prints the whole list (with *print_list*) before and after removing (*erase* function) a triplet.