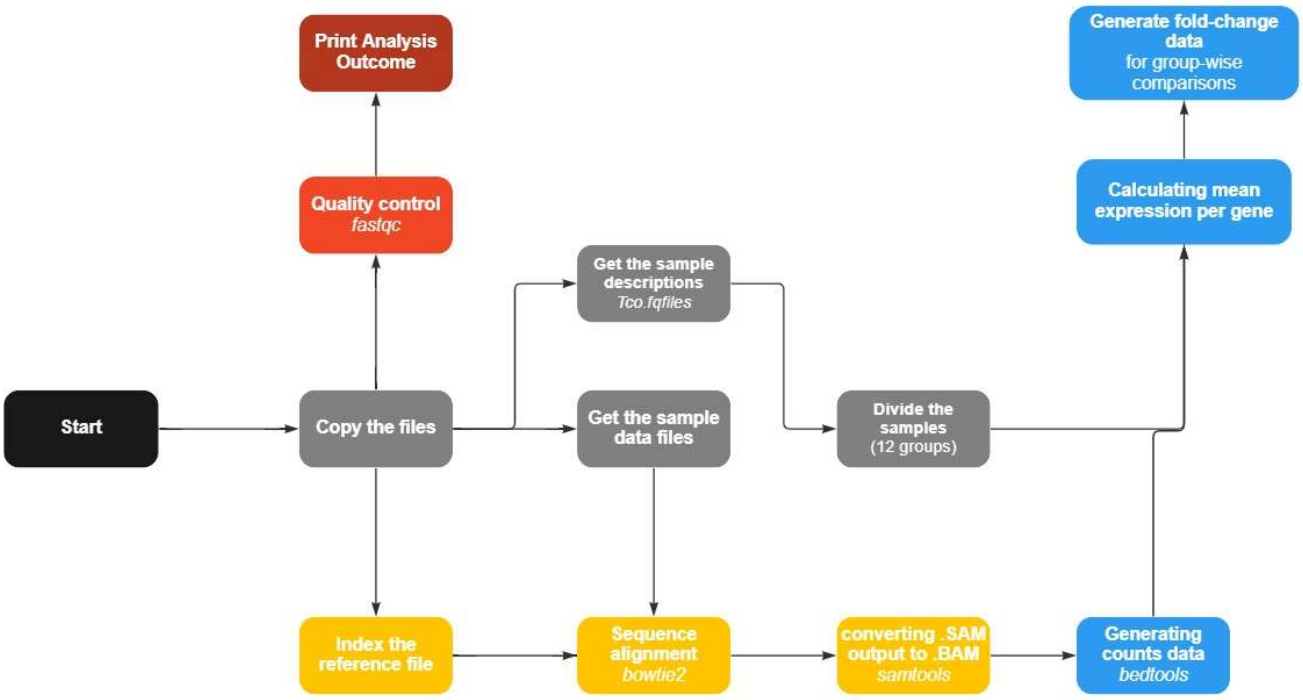
1. **Github ICA1 repository**: https://github.com/B226659-2022/ICA1
2. ***ccrypt* encryption key**: BB226659P123
3. **Overview flowchart:**



4. **Programming parameters and reasoning.**
    1. *cp* – because this function is a bit slow and verbatim options poor, I attempted to use 'scp'. It can be more secure as well as faster and provide a nicer experience for the user
    2. *fastqc –extract* – I used –extract to avoid running a loop that would unzip the results later on (it took very long), so that was a nice way to save time as well as keep code tidier.
    For quite a few functions I used a parallel extension (GNU), which was already installed on our servers. When looking for ways to speed up the processes I came across it on the internet and this way provided me with a 120% faster process
    3. I run a relatively long *while true* loop. I used per base sequence quality because it is the most valuable information for us as it tells us about the certainty of each base being correct, I also added a prompt for "Overrepresented sequences". The latter one can also be a very important check, as it could point out if samples have been contaminated etc. The point of running two prompts was to just show how easily we can create new ones for different analyses.
    4. *Bowtie2-build* Unfortunately did not find a way to multithread or parallel that function, which can make sense if we think about working with just one file.

5. *Bowtie2*. I used the operand *–very-fast-local* alignment, as well as input the number of threads to run the function on, which did speed up the function. I piped the outcome straight to *samtools – view* to avoid saving such files on the local machine, but rather keep it in RAM. There was also an interesting operant in the manual for SAM (--rg-id), which theoretically could help as a group the samples beforehand. Due to lack of time, I could not make it work, so had to continue without it
6. *bedtools intersect -c.* Intersect check where and how many times our samples have shared similar features that overlapped with the .bed file containing genome locations.
   -c option adds a tab-separated column with the number of reads, which is just what we needed. I also decided to run this function in parallel and it made the function run so much faster. Instead of 5-10min, in parallel, it only takes around 30-60 seconds and makes use of all the resources available
7. In part 5 I did not manage to find the correct way of doing the task. I used a multitude of various *for* and *while* loops while trying to join information from separate tables using awk and although I did achieve the result after countless hours of trial and error it is not a recommended approach as it is 'messy', slow and confusing.
8. In part 6 I would most likely need to create a much nicer to-view table, with heading and indexed rows. I might need to make use of data frames to get the fold change between different group-wise comparisons. Because we are left with plenty of vales of 0 in our mean expression of the counts per gene, I would most likely, need to get rid of 0's and change them into just very small values to avoid dividing by 0.
   I would make the output interactive, by giving the user a choice of 8 different group-wise comparisons, because any of them might convey a piece of very useful information which the user could call by clicking a corresponding number on the keyboard.

5. **What things does the user must do to make things work?**
   Nothing! The script will run itself and download raw data files into the user's current directory. In case the user prefers to run the script somewhere else, there is a very simple way to do that. I included a variable called "destination" which is originally set to 'pwd' but can be changed to whatever is desired.

   The user will also be prompted to interact with the script (3 prompts) to choose what sort of quality controls he/she might want to run to be certain of the quality of the data.

6. **Difficulties I have come across.**
   Step 5 was by far the most challenging part. I think at first, I struggled with even understanding what sort of outcome I am looking for. FAQ page helped me with that, so soon I could better imagine which direction to take.

   In the end, it all came down to one problem. It is extremely difficult to merge more than 2 tables in Unix. There is a very neat function "join", but it can only take in two arguments. Because, I did not know how many files, clones, and replicates we are going to need it was impossible to use

join. After scavenging through the internet, it seemed like there is no other way to solve the issue of joining the tables with so many variables and unknowns. I wrote a script in SQL, which was a lot easier even though my knowledge of SQL is very limited, but from what I understand we were not supposed to use it. In the end, I managed to produce the summary files containing the mean expression with the help of a paste function and an interesting way of using loops to print file names in the format that paste takes in as arguments. Unfortunately, that was the reason why I failed to complete task 6.

I also struggled with speeding up the process of indexing the reference file. If there is a way, I would very much like to know. Thank you