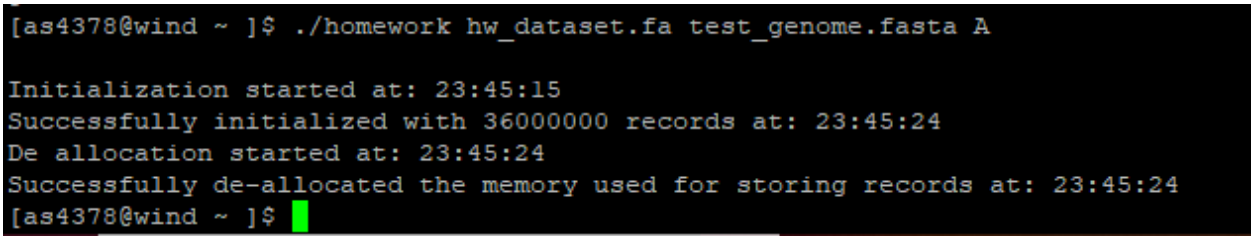
Solutions

A. For initialization I have used a linked list data structure to store the lines. The declaration is:

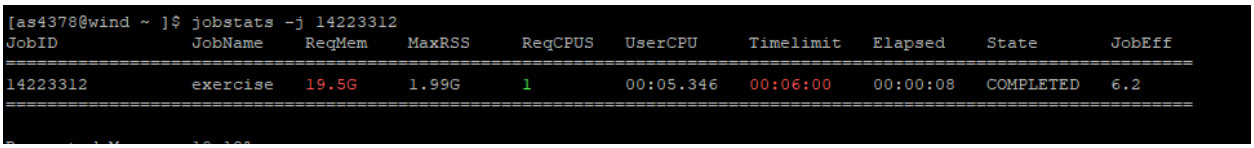
```
struct Node {
        char line[NucleoLineSize]; //equals to 51 represents the length of line
        struct Node* next; // pointer to next node
};
```

The CPU time to initialize 36 million reads is as shown:



From the figure we can see that initialization started at 23:45:15 and ended at 23:35:24. So it took about **9 seconds** for initialization.

The memory utilization can be seen under MaxRSS when running the job through srun.



We can see that it took about 1.99GB memory to store the linked list containing records.

B. To de-allocate the linked -list data structure we need to make sure to clear memory consumed by each node in the list. For this we will have to loop through the linked list and use **delete[]** to clear memory of each node.

The actual time taken is as shown:

```
[as4378@wind ~ ]$ ./homework hw_dataset.fa test_genome.fasta B

Initialization started at: 23:46:29
Successfully initialized with 36000000 records at: 23:46:39
De allocation started at: 23:46:39
Successfully de-allocated the memory used for storing records at: 23:46:39
[as4378@wind ~ ]$
```

From the figure we can see de-allocation started at 23:46:39 and ended at the same time. This means it is taking less than a second. This is because when we are de-allocating then we are only removing pointers i.e. releasing the storage space allocated to it.

C. To perform deep copy, we first have to initialize an object then pass this as an argument to the copy constructor. In this case I have initialized the first object with 36 million reads from the data set and also the data from genome sequence which is shown in the figure.

```
[as4378@wind ~ ]$ ./homework hw_dataset.fa test_genome.fasta C
Initialization started at: 23:52:38
Successfully initialized with 36000000 records at: 23:52:47
Initializing genome data started at:23:52:47
Successfully initialized with 5227244 records of 50 mers at:23:52:50
deep copy started at:23:52:50
deep copy completed at:23:52:54
 De allocation started at: 23:52:54
Successfully de-allocated the memory used for storing records at: 23:52:55
 De allocation started at: 23:52:55
Successfully de-allocated the memory used for storing records at: 23:52:56
[as4378@wind ~ ]$
```

We see that deep copy started at 23:52:50 and ended at 23:52:54. Therefore it took about 4 seconds to perform the deep copy. To perform deep copy we have to create linked list for both the data set and genome sequence for the second object as well and then loop through the respective lists in the first object to copy the data in the lists of second

object. If we compare the time between initialization of first object and time taken for deep copy we can see that it takes comparatively less time to perform deep copy. This is because when we are preforming deep copy then we don't need to read records from file and the first object is already there in memory, so it is very fast.

D. To implement the search functionality efficiently. I first sorted the records in the dataset using quick sort algorithm. For this I implemented a string comparison function similar to strcmp in string library. Then I created an array of character pointers (char* []) pointing to the nodes in the linked list.
Then I sorted the array. If one string is needs to be swapped
with other we will swap the pointers pointing to that strings in array of pointers (char* v[]). This way we can save a lot of copying operations.

To perform search, I implemented a binary search algorithm which returns the index in the sorted array of character pointers if a match is found otherwise it returns -1 which corresponds to the NULL pointer. The time complexity for this algorithm is O (log n). The results are as shown:

```
[as4378@wind ~ ]$ ./homework hw_dataset.fa test_genome.fasta D
Initialization started at: 11:39:17
Successfully initialized with 36000000 records at: 11:39:36
Sorting started at: 11:39:36
Sorted the dataset at: 11:41:16
Performing search for sequence 1 CTAGGTACATCCACACACAGCAGCGCATTATGTATTTATTGGATTTATTT
Found at position 16869981 in sorted dataset
Performing search for sequence 2 CTAGGTACATCCACACACAGCAGCGCATTATGTATTTATTGGATTTATTT
Not found
Performing search for sequence 3 CTAGGTACATCCACACACAGCAGCGCATTATGTATTTATTGGATTTATTT
Not found
Performing search for sequence 4 CTAGGTACATCCACACACAGCAGCGCATTATGTATTTATTGGATTTATTT
Found at position 16569256 in sorted dataset
Performing search for sequence 5 CTAGGTACATCCACACACAGCAGCGCATTATGTATTTATTGGATTTATTT
Found at position 5143294 in sorted dataset
 De allocation started at: 11:41:16
Successfully de-allocated the memory used for storing records at: 11:41:16
[as4378@wind ~ ]$
```

We see that sequence 2 and 3 are not found in the data set.

2(A). To implement this, we need to form all the overlapping 50 mers sequence. This can be easily done by implementing a O(n) time algorithm. We start reading the file character by character and store them in a temporary array of characters. When the character count becomes 50 then we create a node and copy the characters to its **line** member. Then I have implemented a re-arrange function that will shift each character one place to the left in the temporary array. From here on as soon we read the next character we form a new node and add to the linked list. Since the length of temporary array is fixed (50) the rearrange takes 50 operations and we need to perform this approximately **n** times if the total number of characters in the file are **n**. Therefore, total number of operations are approximately **50*n**. This gives us a time complexity of **O(n).**

```
[as4378@wind ~ ]$ ./homework hw_dataset.fa test_genome.fasta E

Initializing genome data started at:0:17:49
Successfully initialized with 5227244 records of 50 mers at:0:17:52
De allocation started at: 0:17:52
Successfully de-allocated the memory used for storing records at: 0:17:52
[as4378@wind ~ ]$
```

From the figure we can see that initialization started at 0:17:49 and ended at 0:17:52. So it took about **3 seconds.** Also, we found that total **5227244** 50 mers sequence are found.

2(B). The results of querying all the 50 mers sequence in part 2(A) are as shown:

```
[as4378@wind ~ ]$ ./homework hw_dataset.fa test_genome.fasta F

Initialization started at: 0:18:40
Successfully initialized with 36000000 records at: 0:18:49
Sorting started at: 0:18:49
Sorted the dataset at: 0:19:42
Initializing genome data started at:0:19:42
Successfully initialized with 5227244 records of 50 mers at:0:19:44
Search for genomic sequences started at: 0:19:44
Search ended at: 0:19:56
Total 1837503 50 mers sequences match were found
De allocation started at: 0:19:56
Successfully de-allocated the memory used for storing records at: 0:19:57
[as4378@wind ~ ]$
```

We see that a total of **1837503** sequence match are found in the dataset. The search started at 0:19:44 and ended at 0:19:56. So it took about **12** seconds.