Solutions

A. For this problem for 1 million reads the memory consumed was not enough. So, I used 10 million instead of 1 million and estimated accordingly.
Time spent on reading first 10 million data sets from the file is as shown in the below table:

```
[as4378@wind ~ ]$ ./homework hw_dataset.fa A
Initialization started at: 12:46:1
Successfully initialized with first 10 million reads at: 12:46:6
De allocation started at: 12:46:6
Successfully de-allocated the memory used for storing records at: 12:46:6
[as4378@wind ~ ]$
```

Each line of the file is having a fixed length of characters (header lines have at most 40 characters and genomic sequences are of 50 characters each) so we can denote the time taken to read a line and copying it to array as O(c) where 'c' denotes the constant no of steps taken to perform this operation which translates to O (1) in terms of complexity. If we perform this kind of read operation 'n' times, then we are spending 'cn' operations overall. Therefore, the time complexity translates to O(cn). Now for each dataset we must perform 2 reads, 1 for header line and 1 for genomic sequence. So, for n datasets we are performing '2cn' operations. From time spent on 10 million datasets we can calculate our c as **2cn = 5seconds. Therefore, c = 2.5*10^ (-7).** This would estimate time taken for 36 million as: **2*2.5*10^ (-7) *36*10^6 = 18 seconds**.

The memory requirements are as shown in jobstats when running the job on monsoon as below under **MaxRSS** column:

```
[as4378@wind ~ ]$ jobstats -j 14108200
JobID            JobName    ReqMem    MaxRSS     ReqCPUS   UserCPU     Timelimit   Elapsed    State       JobEff
================================================================================================================
14108200         exercise   9.77G     1.34G      1         00:02.136   00:06:00    00:00:05   COMPLETED   7.56
================================================================================================================

Requested Memory: 13.73%
Requested Cores : -
Time Limit      : 01.39%
=====================
Efficiency Score: 7.56
=====================
```

This would estimate the memory for 36 million as (1.34/10) *36 GB as complexity of memory is O(n), where n is the number of records read. This comes out to be **4.824GB** of memory.

B. Following shows the actual results of executing read for 36 million data sets.

```
[as4378@wind ~ ]$ ./homework hw_dataset.fa B
Initialization started at: 0:6:53
Successfully initialized with 36 million reads at: 0:7:9
De allocation started at: 0:7:9
Successfully de-allocated the memory used for storing records at: 0:7:11
[as4378@wind ~ ]$
```

Therefore, it took **16 seconds (7:09 – 6:53)** to initialize with 36 million reads. This discrepancy might be due to various factors like CPU availability, time taken by CPU to allocate requested memory which depends on the available memory at time of allocation. In this case the time taken is fairly close to the predicted time in part A.

The memory requirements are as shown in jobstats when running the job on monsoon as below under **MaxRSS** column:

```
[as4378@wind ~ ]$ jobstats -j 14107967
JobID            JobName    ReqMem    MaxRSS     ReqCPUS   UserCPU     Timelimit   Elapsed    State       JobEff
================================================================================================================
14107967         exercise   9.77G     4.83G      1         00:07.119   00:06:00    00:00:22   COMPLETED   27.78
================================================================================================================

Requested Memory: 49.44%
Requested Cores : -
Time Limit      : 06.11%
======================
Efficiency Score: 27.77
======================
```

This value is also close to what was approximated in part A.

C. The statistics are as shown by following code output:

```
[as4378@wind ~ ]$ ./homework hw_dataset.fa C
Initialization started at: 0:8:35
Successfully initialized with 36 million reads at: 0:8:50
The total number of unique sequence fragments are: 36220411
The number of reads for each data set are as follows:
DataSet 1: 3970133
DataSet 2: 3789286
DataSet 3: 3882948
DataSet 4: 3978555
DataSet 5: 3967963
DataSet 6: 3693531
DataSet 7: 3966962
DataSet 8: 3967194
DataSet 9: 3966614
DataSet 10: 3966181
DataSet 11: 3975247
DataSet 12: 3924721
DataSet 13: 3913010
DataSet 14: 3957405
The total number of characters present in the data set are as follows:
A: 493102372
C: 406639890
G: 408544523
T: 489180159
De allocation started at: 0:9:32
Successfully de-allocated the memory used for storing records at: 0:9:33
[as4378@wind ~ ]$
```

D. Time taken to de – allocate the memory is as shown:

```
[as4378@wind ~ ]$ ./homework hw_dataset.fa D
Initialization started at: 0:12:50
Successfully initialized with 36 million reads at: 0:13:4
De allocation started at: 0:13:4
Successfully de-allocated the memory used for storing records at: 0:13:6
[as4378@wind ~ ]$
```

When implementing delete function we had to make sure that memory pointed by pointers to individual strings must be deleted as well. So, we will have to loop through this character pointer array to delete the memory and then delete the pointer to this array. Hence, we will have to loop 36 million times to de-allocate the memory for strings and then de-allocate the memory consumed by this pointer array.
From the above figure we can see that de allocation started at 13:4 (13^{th} minute and 4^{th} second) and ended at 13:6. Hence, it took **2 seconds**. The time taken is comparatively less than the size of memory to be de-allocated because in de-allocation only pointer reference is removed from the memory location which takes very small time. So, even though we are looping 36 million times, it takes less time for de allocating.

E. For sorting the genomic sequences, I have implemented an 'out of place' sorting algorithm using quick sort as base algorithm. First we create an array of character pointers (char *v[])each element of which points to beginning of string representing our genomic sequences. Therefore, the length of this array is equal to number of genomic sequences to be sorted.
Then, I have used Quick Sort and passed this array of character pointers as an argument. Using the help of 'strcmp' function from string library we can compare 2 strings by passing character pointer to strings as an argument. 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.
Time complexity of QuickSort is O (n^2) but that is only when array is already sorted in increasing or decreasing order or all elements are same. In other cases, it is O (n logn). In our case for comparing strings we are using strcmp function, since our strings are of constant size (50), the comparison function is going to take constant number of steps which will translate to time complexity of O (1). Therefore, for n lines our time complexity will be O (n logn).
After sorting the array of character pointers, we copy all the strings pointed by these pointers to new array. This will take time complexity of O(n) for n lines. Hence, the total time complexity of the algorithm is O (n logn) + O(n) = O(nlogn)

Following is the result of executing it on 36 million reads:

```
[as4378@wind ~ ]$ ./homework hw_dataset.fa E
Initialization started at: 0:13:53
Successfully initialized with 36 million reads at: 0:14:9
Sorting started at: 0:14:9
Sorted the dataset at: 0:15:16
First 10 lines after sorting are as follows:
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAG
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAT
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACT
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAGA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAGG
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAATA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAGAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAGAG
De allocation started at: 0:15:29
Successfully de-allocated the memory used for storing records at: 0:15:31
[as4378@wind ~ ]$
```

From the figure we can calculate the time taken for sorting as (15:16 – 14:9) = 1:07
i.e. **1 minute and 7 seconds**