Roll - 200050006
Ques - 1:

In this question , we had to sort an array whose length is present at the location 0x40000000 and the array itself starts from 0x40000001.

I have used the selection sort method.
In the selection sort technique, the list is divided into two parts. In one part all elements are sorted and in another part the items are unsorted. At first we take the maximum or minimum data from the array. After getting the data (say minimum) we place it at the beginning of the list by replacing the data of first place with the minimum data. After performing the array is getting smaller. Thus this sorting technique is done.

I have assumed maximum size of the array as 59999999 because I have mentioned the starting address as 0x40000001 and ending address as 99999999 for read and write access in memory map ini file.

I have used the R0 register to store the the address of the location of array i.e 0x40000000.
Used the R1 register to actually store the length of array taken from the position in register R0. Then I incremented the R0 register by 1 so that R0 points to the first value of array. And also reduces R1 by 1 so that i could use it for no of iterations in loop as 0 to n-1. So I used R2 to mark the iteration count (i) for the outer loop in selection sort and start it from 0.

Similarly I used the R3 as min_index and R4 as j of the actual code and initialised with i+1.
for2 loop is the inner loop of the selection sort and i have mentioned the stopping condition as j>n-1. When the condition is matched, the inner loop will stop.

I have used the R5 register to store the value of array[j] and R6 register to store the value of array[min_index].

The explanation of the selection sort algorithm is mentioned below.
I have used some references to compare the registers to the variables in actual c++ code in order to better explain the code as well as the algorithm. The code whose variables I have referred to is also mentioned below just after the explanation.

Explanation of selection sort :

The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from the unsorted part and putting it at the beginning. The algorithm maintains two subarrays in a given array.

1) The subarray is already sorted.
2) Remaining subarray which is unsorted.

In every iteration of the selection sort, the minimum element (considering ascending order) from the unsorted subarray is picked and moved to the sorted subarray.
This sorting algorithm is an in-place comparison-based algorithm in which the list is divided into two parts, the sorted part at the left end and the unsorted part at the right end. Initially, the sorted part is empty and the unsorted part is the entire list.

The smallest element is selected from the unsorted array and swapped with the leftmost element, and that element becomes a part of the sorted array. This process continues moving unsorted array boundary by one element to the right.
This algorithm is not suitable for large data sets as its average and worst-case complexities are of O(n2), where n is the number of items.

Actual Selection Sort Function in C++

```
void selectionSort(int *array, int n) {
  int i, j, min_index;
  for(i = 0; i<n-1; i++) {
    min_index = i;   //get index of minimum data
    for(j = i+1; j<size; j++)
      if(array[j] < array[min_index])
        min_index = j;
      //placing in correct position
      //swap(array[i], array[min_index]);
       int temp = array[i];
       array[i] = array[min_index];
       array[min_index] = array[i];
  }
}
```