Approach:

So, the approach followed by our team was straightforward; we derived the basic idea of this solution from the assignment given earlier.

We divided the problem statement into 4 parts-

1) Implement a singly linked list traversal.
2) Implement a singly linked list starting the traversal from a random address.
3) Traversing in both directions sequentially.
4) Calculating the minimum data and address.

The first portion of the problem statement was basic with using a MUX for choosing between the current address and the next address value for the shift register to give to the ROM.

For the second portion, we again used a MUX for choosing between the initial address given by the user and the address value coming as feedback.

For the third portion, we then paid attention to address 255, so now while traversing the address 255 will come 2 times in total. When the 255 address isn't reached a single time, then the adder has to add 1 to the current address to get the address of the next node (given that we are first traversing in the forward direction) and then when 255 is reached the first time after that 2 is to be added for the address of next node hence to accomplish this "counting" of the number of time 255 is reached we used a 2 bit counter.

For the last portion, we used a comparator and a register feedback concept that the comparator will keep on comparing the new data values with the previous values stored in the register and update it if the new value is lesser than the previous value.

After implementing this above approach we were ready with the basic block diagram of the circuit but it was not very efficient; we used various components that were not very necessary for comparing when the address reached 255 we used a comparator, but that could also be implemented using a 3 input or gate because of the fact that if any one of the values in the 3 MSB of the number is 1, then the number is sure to be 255 because all the other addresses are less than 31. Another example of cost-cutting is that wherever we were using a MUX we used 2- 4 bit MUX which was unnecessary because

we were only comparing addresses and addresses were only 5-bit numbers hence we used 1 MUX and a combination of 2 AND and 1 OR gate for the 5^{th} bit.

Challenges faced-

The biggest challenge in the solution was to generate proper clock signals for different components like MUX, registers, etc. For this, we used a half clock, shifted clock, and generation of clock pulses using flip flops and logic gates.

This is the point where we feel we could have been a little more efficient by properly drawing FSM for every clock requirement and seeing if that could be derived from an existing clock source by using logic gates which would have been cheaper as compared to the multiple d flip flops used.

A table showing the total cost of the circuit is given below-

Name of component	Cost of each component	No. used	Total cost of the component
ROM-2732	75	1	75
CLOCK	40	1	40
IC-74157	2	2	4
IC-4008	2	1	2
IC-74179	2	4	8
IC-7485	2	2	4
IC-7474	1	7	7
3 INPUT GATES	0.2	2	0.4
2 INPUT GATES	0.1	18	1.8
		TOTAL	142.2

Hence the total cost of the circuit is 142.2sp