Name : Dhruvanshu Parmar

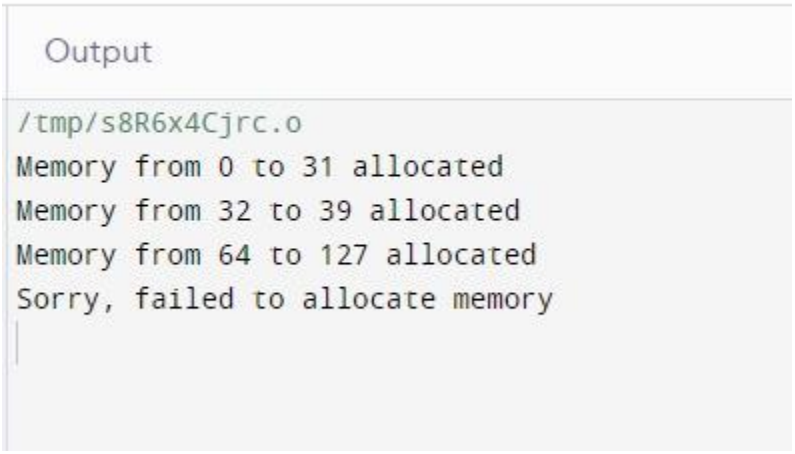
Enrollment ID : AU1940166

Q2b)

Allocation of memory:

The buddy system is executed as follows-A rundown of free nodes, of the relative multitude of various potential powers of 2, is kept up with consistently So on the off chance that all out memory size is 1 MB, marry have 20 free inclines to follow one for squares of size 1 byte, 1 for 2 bytes, next for 4 bytes and so on.At the point when a allocation for a portion comes, we search for the littlest square greater than it. In the event that such a square is found on the free rundown, the portion is done (say, the Allocation is of 27 KB and the free rundown following 32 KB blocks has no less than one component in it), else we navigate the free rundown upwards till we track down a large enough square. Then, at that point, we continue to part it in two squares one for adding to the following free rundown (of more modest size), one to navigate down the tree till we arrive at the objective and return the mentioned memory square to the user. If no such allocation is possible, we simply return null.

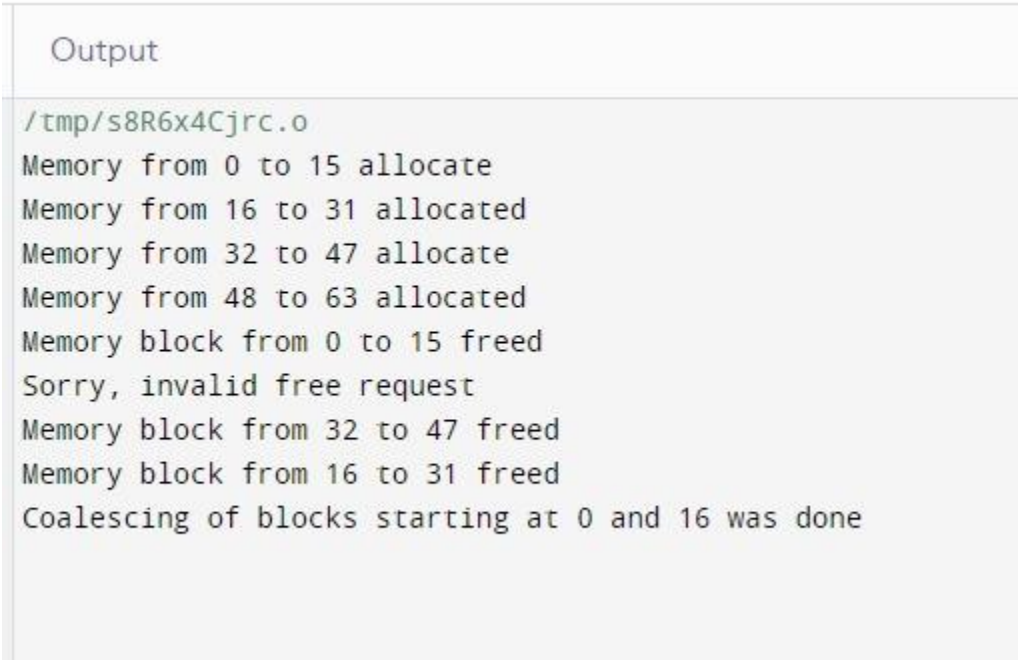
Output:



Deallocation of memory :

The allocation is done via the usage of free lists. Now, for deallocation, we will maintain an extra data structure-a Map with the starting address of segment as key and size of the segment as value and update it whenever an allocation request comes. Now, when a deallocation request comes, we will first check the map to see if it is a valid request. If so, we will then add the block to the free list tracking blocks of their sizes. Then, we will search the free list to see if its *buddy* is free-if so, we will merge the blocks and place them on the free list above them , else we will not coalesce and simply return after that.

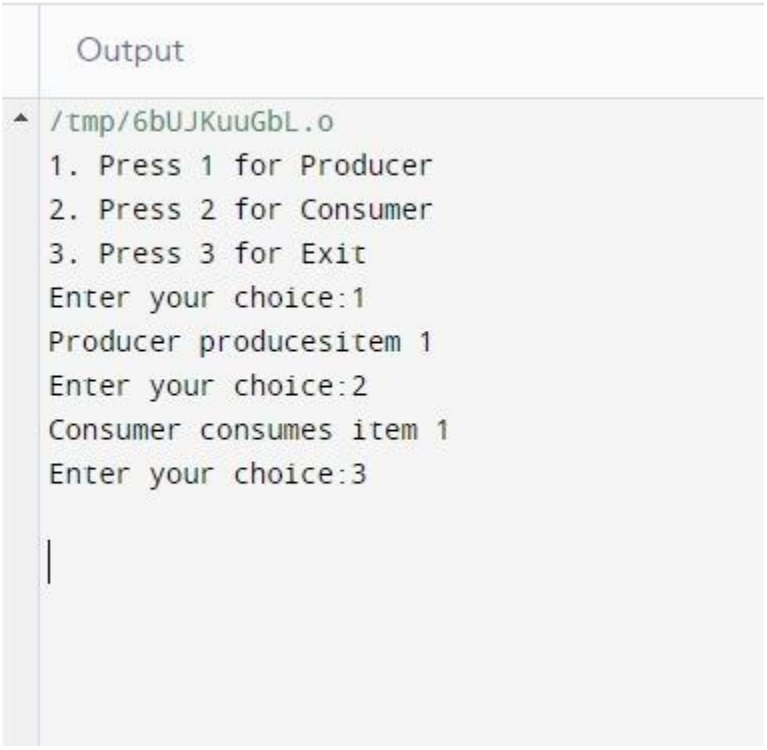
Output:



Q3)

The producer is to either go to sleep or discard data if the buffer is full. The next time the consumer removes an item from the buffer, it notifies the producer, who starts to fill the buffer again. In the same manner, the consumer can go to sleep if it finds the buffer to be empty. The next time the producer puts data into the buffer, it wakes up the sleeping consumer.

Output:


Output
/tmp/6bUJKuuGbL.o
1. Press 1 for Producer
2. Press 2 for Consumer
3. Press 3 for Exit
Enter your choice:1
Producer producesitem 1
Enter your choice:2
Consumer consumes item 1
Enter your choice:3