Name - Parshwa Shah
Roll Number - AU1940263

Question 1 (Multi-threading) -

```
┌──(kali㉿kali)-[/media/…/Parshwa/Programming/AU/endSem]
└─$ gcc -o test Q1.c -pthread

┌──(kali㉿kali)-[/media/…/Parshwa/Programming/AU/endSem]
└─$ ./test
A 1 a B 2 b C 3 c D 4 d E 5 e F 6 f G 7 g H 8 h I 9 i J 10 j K 11 k L 12 l M 13 m N 14 n O 15 o P 16 p Q 17 q R 18 r S 19 s T 20 t U 21 u V 22 v W 23 w X 24 x Y 25 y Z 26 z
```

Here, In the code I have created 3 threads 1 to print A to Z, 2nd to print a to z and 3rd to print 1 to 26. Now all these threads will work at same time. Which actually will create chaos and print all of these out of sync. So now to get all of them in sync I have used mutex and conditional waiting and signaling. What we do here is we have 3 conditional variables. Each of them for 1 thread.

When a thread is willing to work but it's not it's turn to work it'll get wait signal and will be connected to a conditional variable which then will be signaled when ever it's turn comes.

Now in the last part of thread it'll signal the next thread to work and print the next required character.

Question 2 (Buddy's Algorithm) -

```
┌──(kali㉿kali)-[/media/…/Parshwa/Programming/AU/endSem]
└─$ ./a.out
Memory from 0 to 7 allocate
Memory from 16 to 31 allocated
Memory from 32 to 63 allocated
Memory from 64 to 127 allocated
Memory block from 0 to 7 freed
Connecting blocks from 0 and 8 was done
Memory block from 16 to 31 freed
Connecting blocks from 0 and 16 was done
Can't find the target
Memory block from 32 to 63 freed
Connecting blocks from 0 and 32 was done
```

Buddy Memory allocation algorithm first sees the total memory as blocks of 2's power meaning blocks of 2 bytes, 4 bytes, 16 bytes…..

Now what will happen is when you want a bit of memory it'll get nearest big block available in terms of power of 2. And it'll be saved as key value pair showing key will be starting point of allocation and size as value.

Now when you want to deallocate the part you'll need the starting point of the memory and it'll get the size of it and will remove it's if available.

Question 3 (Bonus) -

```
┌──(kali㉿kali)-[/media/…/Parshwa/Programming/AU/endSem]
└─$ gcc -o test Q2.c -pthread

┌──(kali㉿kali)-[/media/…/Parshwa/Programming/AU/endSem]
└─$ ./test
Producer 1: Insert Item 1804289383 at 0
Producer 1: Insert Item 1957747793 at 1
Producer 3: Insert Item 1681692777 at 2
Producer 4: Insert Item 1714636915 at 3
Producer 2: Insert Item 846930886 at 4
Consumer 1: Remove Item 1804289383 from 0
Consumer 1: Remove Item 1957747793 from 1
Producer 1: Insert Item 424238335 at 0
Producer 5: Insert Item 719885386 at 1
Consumer 1: Remove Item 1681692777 from 2
Consumer 2: Remove Item 1714636915 from 3
Producer 3: Insert Item 1649760492 at 2
Producer 4: Insert Item 596516649 at 3
Consumer 1: Remove Item 846930886 from 4
Consumer 3: Remove Item 424238335 from 0
Consumer 4: Remove Item 719885386 from 1
Producer 1: Insert Item 1025202362 at 4
Producer 5: Insert Item 1350490027 at 0
Consumer 2: Remove Item 1649760492 from 2
Consumer 1: Remove Item 596516649 from 3
Producer 2: Insert Item 1189641421 at 1
Consumer 3: Remove Item 1025202362 from 4
Producer 1: Insert Item 2044897763 at 2
Consumer 5: Remove Item 1350490027 from 0
Consumer 5: Remove Item 1189641421 from 1
Producer 4: Insert Item 1102520059 at 3
Consumer 4: Remove Item 2044897763 from 2
Producer 3: Insert Item 783368690 at 4
Producer 5: Insert Item 1967513926 at 0
Producer 2: Insert Item 1365180540 at 1
Consumer 4: Remove Item 1102520059 from 3
Producer 4: Insert Item 1540383426 at 2
Consumer 5: Remove Item 783368690 from 4
Consumer 3: Remove Item 1967513926 from 0
Consumer 4: Remove Item 1365180540 from 1
Producer 3: Insert Item 304089172 at 3
Producer 4: Insert Item 521595368 at 4
Producer 5: Insert Item 1303455736 at 0
```

This is very similar to the first problem I have submitted it does same thing with help of concept of streams and from the stream you can send and receives data in terms of boats.