CSE332 - Operating Systems

End Semester Exam

Submitted by, Axay Ghoghari

Enrollment Number : AU1940269

Description File

Q1 : Print the following Pattern A 1 a B 2 b C 3 c … Y 25 y Z 26 z Using any one of the following concepts

b. Multithreads (Hint: using 3 Threads) (PS: Process Synchronization or Thread Synchronization is key thing for pattern printing).

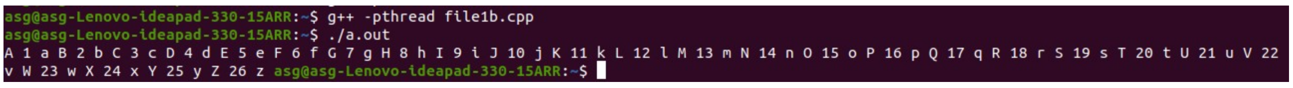
Description :

We will need to include header files :
#include <iostream>
#include <thread>
#include<mutex>
#include<semaphore.h>
#include <unistd.h>

I have made 3 functions, For capital Letters, For small Letters and third for integrate these.

At last in main function we called these functions and used threads. At last I udes join inbuit function for joning capital Letter , small Letter and numbers assosiated with it.

Screenshot :



```
asg@asg-Lenovo-ideapad-330-15ARR:~$ g++ -pthread file1b.cpp
asg@asg-Lenovo-ideapad-330-15ARR:~$ ./a.out
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 asg@asg-Lenovo-ideapad-330-15ARR:~$
```

Q2 : Describe and implement any one of the following.

b. Describe the Buddy’s Algorithm for Memory Allocation and Deallocation along with an example and implement it in C or C++.

Description :

I've done the allocation through the use of free lists. We'll keep an extra data structure for deallocation—a Map with the segment's starting address as key and size as value—and update it whenever an allocation request comes in. When a deallocation request arrives, we'll first look at the map to determine if it's a legitimate request.

If that's the case, we'll add the block to the free list that keeps track of blocks of various sizes. Then we'll check the free list to see if its buddy is available; if so, we'll merge the blocks and position them above them on the free list. Otherwise, we will not coalesce and simply return after tht. I defined two terms-buddyNumber and buddyAddress. Because both the numerator and denominator are powers of 2, this is always an integer. If both of them were generated by the splitting of the same larger block, a block will now be the buddy of another block.

All we have to do now is look for this buddyAddress in the free list (by comparing it to all of the starting addresses in that list), and if it's there, we can coalesce.

Allocation Request is 16 bytes.
No such block exists, so we traverse up and split the 0-127 block into 0-63, 64-127; we add 64-127 to the list tracking 64-byte blocks and pass 0-63 downwards; we split it again into 0-31 and 32-63; we add 32-63 to the list tracking 32-byte blocks and pass 0-31 downwards; we split it again, and 0-15 is returned to the user while 16-31 is added to the free list tracking 16-byte blocks.

Allocation Request is 16 bytes.
Directly memory segment 16-31 will be allocated.

Allocation Request is: 16 bytes
No such block was discovered, so we'll go up to block 32-63 and split it into 32-47 and 48-63; we'll add 48-63 to the list of 16-byte blocks to track, and we'll give 32-47 to the user.

Allocation Request: 16 bytes
Directly memory segment 48-63 will be allocated.

Deallocation Request: StartIndex = 0
Deallocation will take place, but no coalesce will be possible because the buddyNumber is 0 and the buddyAddress is 16 (through the algorithm), both of which are not in the free list.

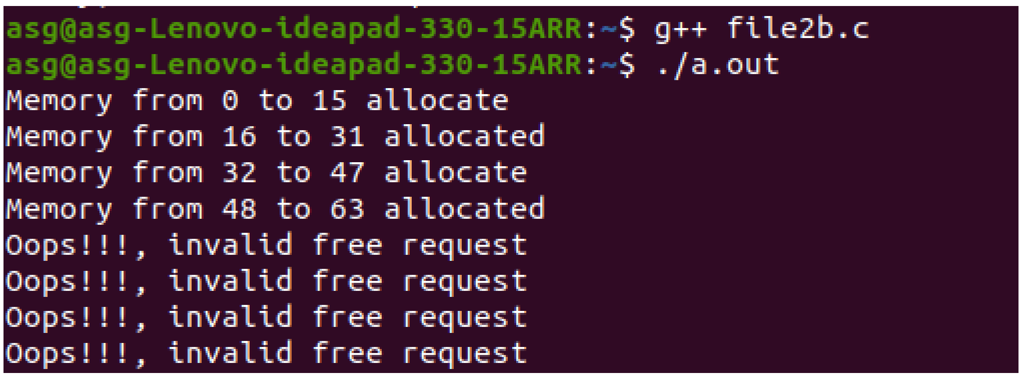
Deallocation Request: StartIndex = 9
This segment was never allocated, hence this request is invalid.

Deallocation Request: StartIndex = 32
Deallocation will take place, but no coalesce will be able due to the buddy's absence. The blocks are numbered 0 and 2 buddy. The blocks' addresses are 16 and 48, respectively, and neither is on the free list.

Deallocation Request: StartIndex = 16

Because the buddyAddress of block 16-31 is 0, which is contained in the free list monitoring 16-byte blocks, deallocation and coalescing of blocks 0-15 and 16-31 will be performed.

Screenshots:



```
asg@asg-Lenovo-ideapad-330-15ARR:~$ g++ file2b.c
asg@asg-Lenovo-ideapad-330-15ARR:~$ ./a.out
Memory from 0 to 15 allocate
Memory from 16 to 31 allocated
Memory from 32 to 47 allocate
Memory from 48 to 63 allocated
Oops!!!, invalid free request
Oops!!!, invalid free request
Oops!!!, invalid free request
Oops!!!, invalid free request
```

Q3 : [Bonus] Describe what is Producer Consumer Problem and its solution in detail using Semaphores and Mutex and implement it in C.

Description :

A fixed-size buffer is used as a queue by both the producer and the consumer. The producer is responsible for generating and storing data in the buffer. It is the consumer's responsibility to look through the data in this buffer one by one.

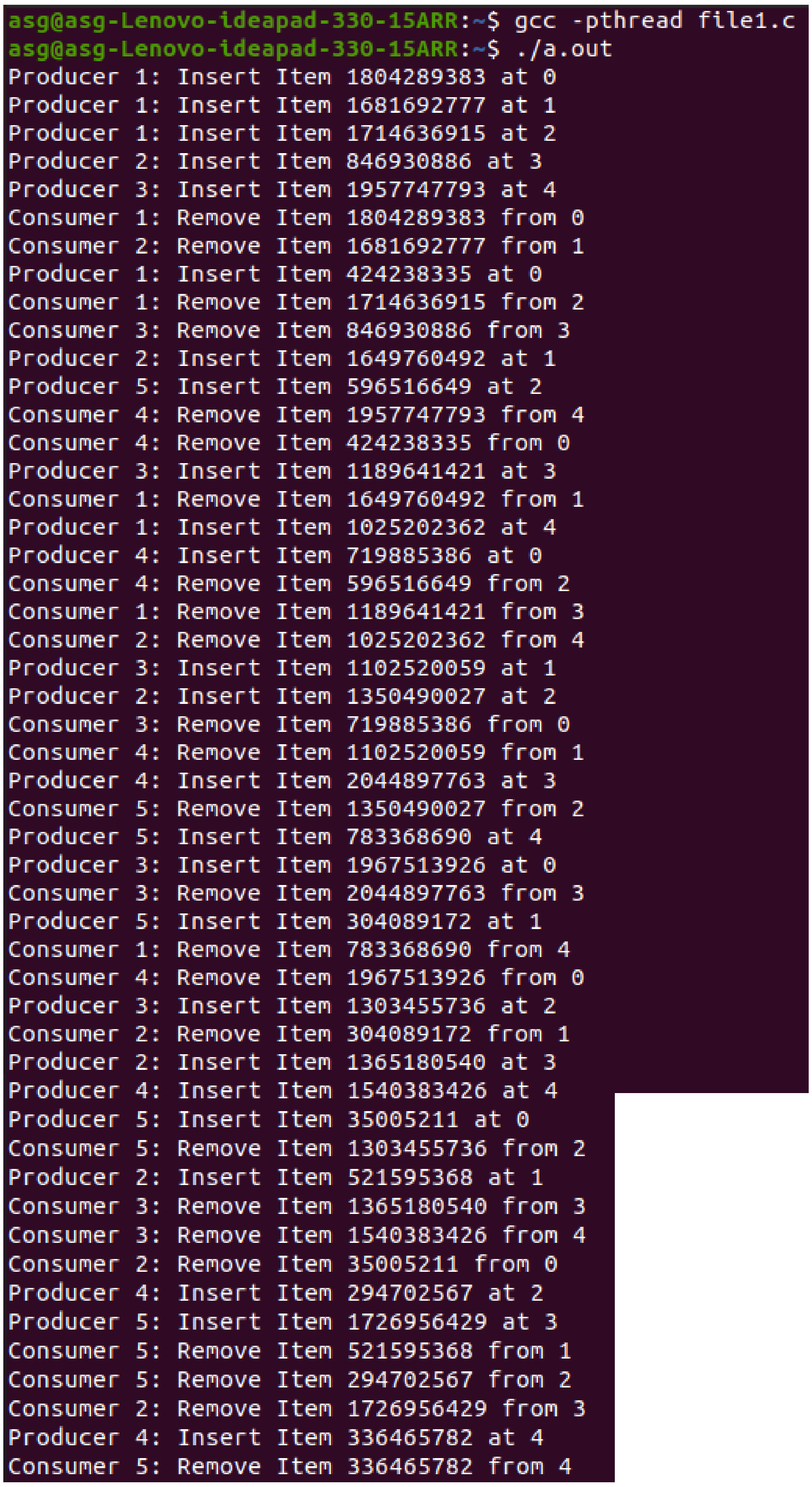
Producer :

```
while(True) {
  // this will produce an item
  wait(Empty); // it will wait/sleep when there are no empty slots available
  wait(Mutex);
  buffer[in] = item
  in = (in+1)%buffer_size;
  signal(Mutex);
  signal(full); // Signal/wake to consumer that buffer has some   data and they can consume now
}
```

Consumer :

```
while(True) {
  wait(full); // it will wait/sleep when there are no full slots
  wait(Mutex);
  item = buffer[out];
  out = (out+1)%buffer_size;
  signal(Mutex);
  signal(Empty); // Signal/wake the producer that buffer slots are emptied and they can produce
more
}
```

Screenshots:



```
asg@asg-Lenovo-ideapad-330-15ARR:~$ gcc -pthread file1.c
asg@asg-Lenovo-ideapad-330-15ARR:~$ ./a.out
Producer 1: Insert Item 1804289383 at 0
Producer 1: Insert Item 1681692777 at 1
Producer 1: Insert Item 1714636915 at 2
Producer 2: Insert Item 846930886 at 3
Producer 3: Insert Item 1957747793 at 4
Consumer 1: Remove Item 1804289383 from 0
Consumer 2: Remove Item 1681692777 from 1
Producer 1: Insert Item 424238335 at 0
Consumer 1: Remove Item 1714636915 from 2
Consumer 3: Remove Item 846930886 from 3
Producer 2: Insert Item 1649760492 at 1
Producer 5: Insert Item 596516649 at 2
Consumer 4: Remove Item 1957747793 from 4
Consumer 4: Remove Item 424238335 from 0
Producer 3: Insert Item 1189641421 at 3
Consumer 1: Remove Item 1649760492 from 1
Producer 1: Insert Item 1025202362 at 4
Producer 4: Insert Item 719885386 at 0
Consumer 4: Remove Item 596516649 from 2
Consumer 1: Remove Item 1189641421 from 3
Consumer 2: Remove Item 1025202362 from 4
Producer 3: Insert Item 1102520059 at 1
Producer 2: Insert Item 1350490027 at 2
Consumer 3: Remove Item 719885386 from 0
Consumer 4: Remove Item 1102520059 from 1
Producer 4: Insert Item 2044897763 at 3
Consumer 5: Remove Item 1350490027 from 2
Producer 5: Insert Item 783368690 at 4
Producer 3: Insert Item 1967513926 at 0
Consumer 3: Remove Item 2044897763 from 3
Producer 5: Insert Item 304089172 at 1
Consumer 1: Remove Item 783368690 from 4
Consumer 4: Remove Item 1967513926 from 0
Producer 3: Insert Item 1303455736 at 2
Consumer 2: Remove Item 304089172 from 1
Producer 2: Insert Item 1365180540 at 3
Producer 4: Insert Item 1540383426 at 4
Producer 5: Insert Item 35005211 at 0
Consumer 5: Remove Item 1303455736 from 2
Producer 2: Insert Item 521595368 at 1
Consumer 3: Remove Item 1365180540 from 3
Consumer 3: Remove Item 1540383426 from 4
Consumer 2: Remove Item 35005211 from 0
Producer 4: Insert Item 294702567 at 2
Producer 5: Insert Item 1726956429 at 3
Consumer 5: Remove Item 521595368 from 1
Consumer 5: Remove Item 294702567 from 2
Consumer 2: Remove Item 1726956429 from 3
Producer 4: Insert Item 336465782 at 4
Consumer 5: Remove Item 336465782 from 4
```