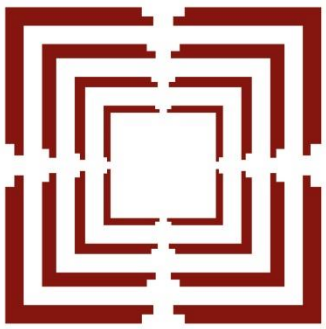

Ahmedabad University

CSE332: Operating System

Prof.: **Priyesh Chaturvedi**

TAs: **Manavkumar Vagrecha, Bhumiti Gohel**

Assignment Submitted by: **Krina Khakhariya**

(AU1940208)

ANSWER 1(CODE)

```
/*Krina Khakhariya
AU1940208*/

/*Question 1. 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
a. Multiprocesses (Hint: using 3 child processes)
b. Multithreads (Hint: using 3 Threads)
(PS: Process Synchronization or Thread Synchronization is key thing
for
pattern printing)*/
#include <iostream>
#include <thread>
#include<mutex>
#include<semaphore.h>
#include <unistd.h>
#define THREAD_NUM 3
using namespace std;

sem_t Small_Letter;
sem_t Capital_Letter;
sem_t num;
//Function for Small Letters
void func_small(){
    for(int i = 1;i<27;i++){
      sem_wait(&num);
        std::cout<<" "<<i<<" ";
    sem_post(&Small_Letter);
    }
}
//Function for Captial Letters
void func_caps(){
```

```
char ch;

    for (ch = 'A'; ch <= 'Z'; ++ch){
     sem_wait(&Capital_Letter);
        std::cout<<ch<<"";
    sem_post(&num);
    }
}
//Function for Numbers
void func_num(){
    char ch;
    for (ch = 'a'; ch <= 'z'; ++ch){
    sem_wait(&Small_Letter);
        std::cout<<ch<<" ";
    sem_post(&Capital_Letter);
    }
}
int main(){
sem_init(&Small_Letter, 0, 0);
sem_init(&Capital_Letter, 0, 1);
sem_init(&num, 0, 0);
  std::thread small_thread,capital_thread,numeric_thread;

  small_thread =  std::thread(func_small);
  capital_thread = std::thread(func_caps);
  numeric_thread = std::thread(func_num);

    capital_thread.join();
    numeric_thread.join();
    small_thread.join();
}
```

OUTPUT - 1

```
PS E:\5th_SEM\OS\Lab_exam\END_EXAM> g++ -pthread Q-1.cpp
PS E:\5th_SEM\OS\Lab_exam\END_EXAM> ./a.exe
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
PS E:\5th_SEM\OS\Lab_exam\END_EXAM>
```

ANSWER 2(CODE)

```cpp
/*Krina Khakhariya
AU1940208*/

/*Question 2(b)
Describe the Buddy's Algorithm for Memory Allocation and
Deallocation along with an example and implement it in C or C++.*/

#include<bits/stdc++.h>
using namespace std;

// size of vector
int size;

vector<pair<int, int>> arr[100000];

map<int, int> mp;

void Buddy_mem(int s)
{
    int n = ceil(log(s) / log(2));

    size = n + 1;
    for(int i = 0; i <= n; i++)
        arr[i].clear();

    arr[n].push_back(make_pair(0, s - 1));
}

void allocate_space(int s)
{
    int x = ceil(log(s) / log(2));
    if (arr[x].size() > 0)
    {
        pair<int, int> temp = arr[x][0];
        arr[x].erase(arr[x].begin());

        cout << "Memory from " << temp.first
            << " to " << temp.second
            << " allocated till here" << "\n";
```

```
        mp[temp.first] = temp.second -
                         temp.first + 1;
    }
    else
    {
        int i;
        for(i = x + 1; i < size; i++)
        {
            // Finding block size greater than given request
            if (arr[i].size() != 0)
                break;
        }
        // If no such block is found i.e., no memory block available
        if (i == size)
        {
            cout << "Oops! Allocated memory failed.\n";
        }
        else
        {
            pair<int, int> temp;
            temp = arr[i][0];

            // Remove first block to splitting it into halves
            arr[i].erase(arr[i].begin());
            i--;

            for(;i >= x; i--)
            {
                pair<int, int> pair1, pair2;
                pair1 = make_pair(temp.first,
                                  temp.first +
                                  (temp.second -
                                  temp.first) / 2);
                pair2 = make_pair(temp.first +
                                  (temp.second -
                                  temp.first + 1) / 2,
                                  temp.second);

                arr[i].push_back(pair1);

                arr[i].push_back(pair2);
                temp = arr[i][0];

                arr[i].erase(arr[i].begin());
            }

            cout << "Memory from " << temp.first
                << " to " << temp.second
                << " allocate" << "\n";
```

```
            mp[temp.first] = temp.second -
                             temp.first + 1;
        }
    }
}

void deallocate_space(int id)
{
    //If address is not found as such
    if(mp.find(id) == mp.end())
    {
        cout << "Oops! Your free request is invalid\n";
        return;
    }

    // Searching the size of block
    int n = ceil(log(mp[id]) / log(2));
    int i, num_buddy, add_buddy;
    arr[n].push_back(make_pair(id,
                           id + pow(2, n) - 1));
    cout << "Memory is getting block from " << id
        << " to "<< id + pow(2, n) - 1
        << " freed\n";

    // Calculate buddy number
    num_buddy = id / mp[id];

    if (num_buddy % 2 != 0)
        add_buddy = id - pow(2, n);
    else
        add_buddy = id + pow(2, n);

    for(i = 0; i < arr[n].size(); i++)
    {
        if (arr[n][i].first == add_buddy)
        {
            //Merging the buddies
            if (num_buddy % 2 == 0)
            {
                arr[n + 1].push_back(make_pair(id,
                id + 2 * (pow(2, n) - 1)));

                cout << "Coalescing of blocks starting from here "
                    << id << " and " << add_buddy
                    << " was done" << "\n";
            }
            else
            {
```

```
                arr[n + 1].push_back(make_pair(
                    add_buddy, add_buddy +
                    2 * (pow(2, n))));

                cout << "Coalescing of blocks starting at "
                    << add_buddy << " and "
                    << id << " was done" << "\n";
            }
            arr[n].erase(arr[n].begin() + i);
            arr[n].erase(arr[n].begin() +
            arr[n].size() - 1);
            break;
        }
    }
    mp.erase(id);
}
int main()
{
    Buddy_mem(128);
    allocate_space(16);
    allocate_space(16);
    allocate_space(16);
    allocate_space(16);
    deallocate_space(0);
    deallocate_space(9);
    deallocate_space(32);
    deallocate_space(16);

    return 0;
}
```

OUTPUT - 2

```
[Running] cd "e:\5th_SEM\OS\Lab_exam\END_EXAM\" && g++ Q-2.cpp -o Q-2 && "e:\5th_SEM\OS\Lab_exam\END_EXAM\"Q-2
Memory from 0 to 15 allocate
Memory from 16 to 31 allocated till here
Memory from 32 to 47 allocate
Memory from 48 to 63 allocated till here
Memory is getting block from 0 to 15 freed
Oops! Your free request is invalid
Memory is getting block from 32 to 47 freed
Memory is getting block from 16 to 31 freed
Coalescing of blocks starting at 0 and 16 was done
```

```
[Done] exited with code=0 in 69.497 seconds
```

ANSWER 3 (CODE)

```
/* Krina Khakhariya
AU1940208

Q-3 bonus question
Describe what is Producer Consumer Problem and its solution in
detail using Semaphores and Mutex and implement it in C.*/
#include <pthread.h>
#include <semaphore.h>
#include <stdlib.h>
#include <stdio.h>
#define max_items 5 // Maximum items a producer can produce or a
consumer can consume
#define buffer_size 5 // Buffer size

sem_t empty;
sem_t full;
int in = 0;
int out = 0;
int buffer[buffer_size];
pthread_mutex_t mutex;

void *producer_func(void *prod_no)
{
    int items;
    for(int i = 0; i < max_items; i++) {
        items = rand(); // Random item will be produce here
        sem_wait(&empty);
        pthread_mutex_lock(&mutex);
        buffer[in] = items;
        printf("Producer %d: Insert Item %d at %d\n", *((int
*)prod_no),buffer[in],in);
        in = (in+1)%buffer_size;
        pthread_mutex_unlock(&mutex);
        sem_post(&full);
    }
}
void *consumer_func(void *cons_no)
{
    int items;
    for(int i = 0; i < max_items; i++) {
        sem_wait(&full);
        pthread_mutex_lock(&mutex);
        int item = buffer[out];
```

```
        printf("Consumer %d: Remove Item %d from %d\n",*((int
*)cons_no),items, out);
        out = (out+1)%buffer_size;
        pthread_mutex_unlock(&mutex);
        sem_post(&empty);
    }
}

int main()
{

    pthread_t prod[5],cons[5];
    pthread_mutex_init(&mutex, NULL);
    sem_init(&empty,0,buffer_size);
    sem_init(&full,0,0);

    int array[5] = {1,2,3,4,5}; //Giving number to consumer and
producer

    for(int i = 0; i < 5; i++) {
        pthread_create(&prod[i], NULL, (void *)producer_func, (void
*)&array[i]);
    }
    for(int i = 0; i < 5; i++) {
        pthread_create(&cons[i], NULL, (void *)consumer_func, (void
*)&array[i]);
    }

    for(int i = 0; i < 5; i++) {
        pthread_join(prod[i], NULL);
    }
    for(int i = 0; i < 5; i++) {
        pthread_join(cons[i], NULL);
    }

    pthread_mutex_destroy(&mutex);
    sem_destroy(&empty);
    sem_destroy(&full);

    return 0;

}
```

OUTPUT

```
[Running] cd "e:\5th_SEM\OS\Lab_exam\END_EXAM\" && gcc Q-3.c -o Q-3
&& "e:\5th_SEM\OS\Lab_exam\END_EXAM\"Q-3
Producer 1: Insert Item 41 at 0
```

```
Producer 1: Insert Item 18467 at 1
Producer 1: Insert Item 6334 at 2
Producer 1: Insert Item 26500 at 3
Producer 1: Insert Item 19169 at 4
Consumer 1: Remove Item 0 from 0
Consumer 1: Remove Item 0 from 1
Consumer 1: Remove Item 0 from 2
Consumer 1: Remove Item 0 from 3
Consumer 1: Remove Item 0 from 4
Producer 2: Insert Item 41 at 0
Producer 2: Insert Item 18467 at 1
Producer 3: Insert Item 41 at 2
Producer 4: Insert Item 41 at 3
Producer 5: Insert Item 41 at 4
Consumer 2: Remove Item 0 from 0
Consumer 2: Remove Item 0 from 1
Consumer 2: Remove Item 0 from 2
Consumer 2: Remove Item 0 from 3
Consumer 2: Remove Item 0 from 4
Producer 2: Insert Item 6334 at 0
Producer 2: Insert Item 26500 at 1
Producer 3: Insert Item 18467 at 2
Producer 4: Insert Item 18467 at 3
Producer 5: Insert Item 18467 at 4
Consumer 3: Remove Item 0 from 0
Consumer 3: Remove Item 0 from 1
Consumer 3: Remove Item 0 from 2
Consumer 3: Remove Item 0 from 3
Consumer 3: Remove Item 0 from 4
Producer 2: Insert Item 19169 at 0
Producer 3: Insert Item 6334 at 1
Producer 3: Insert Item 26500 at 2
Producer 4: Insert Item 6334 at 3
Producer 5: Insert Item 6334 at 4
Consumer 4: Remove Item 0 from 0
Producer 3: Insert Item 19169 at 0
Consumer 4: Remove Item 0 from 1
Consumer 4: Remove Item 0 from 2
Producer 4: Insert Item 26500 at 1
Producer 5: Insert Item 26500 at 2
Consumer 4: Remove Item 0 from 3
Producer 4: Insert Item 19169 at 3
Consumer 4: Remove Item 0 from 4
Producer 5: Insert Item 19169 at 4
Consumer 5: Remove Item 0 from 0
Consumer 5: Remove Item 0 from 1
Consumer 5: Remove Item 0 from 2
Consumer 5: Remove Item 0 from 3
Consumer 5: Remove Item 0 from 4
```

```
[Done] exited with code=0 in 82.207 seconds
```