Operating System Practical Exam
23rd November, 2020

Name: Jay Mehta
Section - 2

Code-1:

```
#include<stdio.h>
#include<stdlib.h>
#include<pthread.h>
#include<semaphore.h>

sem_t sema;
int location = 0;

void *num_function(void *arg){
    int i;
    for(i=0;i<26;i++){
        sem_wait(&sema);
        if(location%3 != 1){
            i--;
        }else{
            printf("%d ",1+i);
            location++;
        }
        sem_post(&sema);
    }
    pthread_exit(NULL);
}

void *capital_function(void *arg){
    int j;
    for(j=0;j<26;j++){
        sem_wait(&sema);
        if(location%3 != 0){
            j--;
        }else{
            printf("%c ",'A'+j);
            location++;
        }
        sem_post(&sema);
    }
    pthread_exit(NULL);
}

void *lowercase_function(void *arg){
    int k;
    for(k=0;k<26;k++){
        sem_wait(&sema);
        if(location%3 != 2){
            k--;
        }else{
            printf("%c ",'a'+k);
            location++;
        }
        sem_post(&sema);
    }
    pthread_exit(NULL);
}

int main(){


    pthread_t tid1, tid2, tid3;
    char start1 = 'A';
    char start2 = 'a';
    sem_init(&sema, 0, 1);
    pthread_create(&tid1,NULL,capital_function,NULL);
    pthread_create(&tid2,NULL,num_function,NULL);
    pthread_create(&tid3,NULL,lowercase_function,NULL);
    pthread_join(tid1,NULL);
    pthread_join(tid2,NULL);
    pthread_join(tid3,NULL);
    sem_destroy(&sema);
    return 0;
}
```

Ans-2

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


int store;


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

map<int, int> final;

void BuddyAlgo(int s)
{

     int n = ceil(log(s) / log(2));

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


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

void allocation(int t)
{


     int n = ceil(log(t) / log(2));
     if (final[n].store() > 0)
     {
          pair<int, int> t = final[n][0];


          final[n].erase(final[n].begin());
          cout << "Memory from " << t.first
               << " to " << t.second << "has been
allocated"
               << "\n";

y
          store[t.first] = t.second -
                             t.first + 1;
     }
     else
     {
          int i;
          for(i = n + 1; i < store; i++)
          {


               if(final[i].store() != 0)
                    break;
          }


          if (i == store)
          {
               cout << "Error in memory allocation \n";
          }


          else
          {
               pair<int, int> t;
```

```
        t = final[i][0];


        final[i].erase(final[i].begin());
        i--;

        for(; i >= n; i--)
        {

            pair<int, int> first, second;
            first = make_pair(t.first,
                              t.first +
                              (t.second -
                              t.first) / 2);
            second = make_pair(t.first +
                              (t.second -
                              t.first + 1) / 2,
                              t.second);

            final[i].push_back(first);


            final[i].push_back(second);
            t = final[i][0];


            final[i].erase(final[i].begin());
        }
        cout << "Memory from " << t.first
            << " to " << t.second
            << " has been allocated" << "\n";

        store[t.first] = t.second -
                          t.first + 1;
    }
}
```

```
}


void deallocation(int s)
{


     if(final.find(s) == final.end())
     {
          cout << "Invalid request\n";
          return;
     }



     int n = ceil(log(final[s]) / log(2));

     int i, b_number, b_address;


     store[n].push_back(make_pair(s,
                              s + pow(2, n) - 1));
     cout << "Memory block from " << s
          << " to "<< s + pow(2, n) - 1
          << "has been freed\n";

     b_number = s / final[s];

     if (b_number % 2 != 0)
          b_address = s - pow(2, n);
     else
          b_address = s + pow(2, n);

     for(i = 0; i < store[n].store(); i++)
     {
        if (store[n][i].first == b_address)
        {


            if (b_number % 2 == 0)
            {
                store[n + 1].push_back(make_pair(s,
                s + 2 * (pow(2, n) - 1)));

                cout << "Coalescing of blocks starting
from "
                    << s << " and " << b_address
                    << " was done" << "\n";
            }
            else
            {
                store[n + 1].push_back(make_pair(
                    b_address, b_address +
                    2 * (pow(2, n))));

                cout << "Coalescing of blocks starting
from "
                    << b_address << " and "
                    << s << " was done" << "\n";
            }
            store[n].erase(store[n].begin() + i);
            store[n].erase(store[n].begin() +
            store[n].store() - 1);
            break;
        }
    }

    final.erase(s);
}

int main()
```

```
{

    BuddyAlgo(2048);
    allocation(16);
    allocation(32);
    allocation(16);
    allocation(9);
    deallocation(0);
    deallocation(9);
    deallocation(32);
    deallocation(16);

     return 0;
}
```

Ans-3

```
#include <stdlib.h>
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>



#define MaxItems 5
#define BufferSize 5

sem_t empty_semaphore;
sem_t full_semaphore;
int up = 0;
int down = 0;
int buffer[BufferSize];
pthread_mutex_t mutex;

void *pro_function(void *pno)
{
    int item;
    for(int i = 0; i < MaxItems; i++) {
        item = rand();
        sem_wait(&empty_semaphore);
        pthread_mutex_lock(&mutex);
        buffer[up] = item;
        printf("Producer %d: Insert Item %d at %d\n", *((int 
*)pno),buffer[up],up);
        up = (up+1)%BufferSize;
        pthread_mutex_unlock(&mutex);
        sem_post(&full_semaphore);
    }
}
void *cons_function(void *cno)
{
    for(int i = 0; i < MaxItems; i++) {
        sem_wait(&full_semaphore);
        pthread_mutex_lock(&mutex);
        int item = buffer[down];
        printf("Consumer %d: Remove Item %d from %d\n",*((int
*)cno),item, down);
        down = (down+1)%BufferSize;
        pthread_mutex_unlock(&mutex);
        sem_post(&empty_semaphore);
    }
}

int main()
{

    pthread_t first[5],second[5];
    pthread_mutex_init(&mutex, NULL);
    sem_init(&empty_semaphore,0,BufferSize);
    sem_init(&full_semaphore,0,0);

    int arr[5] = {1,2,3,4,5};

    for(int i = 0; i < 5; i++) {
        pthread_create(&first[i], NULL, (void *)pro_function,
(void *)&arr[i]);
    }
    for(int i = 0; i < 5; i++) {
        pthread_create(&second[i], NULL, (void
*)cons_function, (void *)&arr[i]);
    }

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

    pthread_mutex_destroy(&mutex);
    sem_destroy(&empty_semaphore);
    sem_destroy(&full_semaphore);

    return 0;

}
```