Dhaval Chaudhary

AU1940180

End-Semester Practical Exam

Question – 1 (b) Multithreads

CODE:-

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

sem_t smh;

int m = 0;

void *thread_function(void *arg)
{
    int d;
    for (d = 0; d < 26; d++)
    {
        s_wait(&smh);
        if (m % 3 != 1)
        {
            d--;
        }
        else
        {
            printf("%d ", 1 + d);
            m++;
        }
        se_po(&smh);
    }
    pthread_exit(NULL);
}

void *thread_function2(void *arg)
```

```
{
    int d;
    for (d = 0; d < 26; d++)
    {
        s_wait(&smh);
        if (m % 3 != 0)
        {
            d--;
        }
        else
        {
            printf("%c ", 'A' + d);
            m++;
        }
        se_po(&smh);
    }
    pthread_exit(NULL);
}

void *thread_function3(void *arg)
{
    int d;
    for (d = 0; d < 26; d++)
    {
        s_wait(&smh);
        if (m % 3 != 2)
        {
            d--;
        }
        else
        {
            printf("%c ", 'a' + d);
            m++;
        }
        se_po(&smh);
    }
    pthread_exit(NULL);
}

int main()
{

    pthread_t t1, t2, t3;
    char s1 = 'A';
    char s2 = 'a';
    sem_init(&smh, 0, 1);
    pthread_c(&t1, NULL, thread_function2, NULL);
    pthread_c(&t2, NULL, thread_function, NULL);
```

```
    pthread_c(&t3, NULL, thread_function3, NULL);

    pthread_j(t1, NULL);

    pthread_j(t2, NULL);

    pthread_j(t3, NULL);
    sem_destroy(&smh);
    return 0;
}
```

Question – 2 (b) Memory Allocation

CODE:-

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

int sz;


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

map<int, int> m;

void initialize(int sz)
{
    int f = ceil(log(sz) / log(2));
    sz = f + 1;

    for(int z = 0; z <= f; z++)
        f_list[z].clear();

    f_list[f].push_back(makeP(0, sz - 1));
}

void allocate(int sz)
{
    int f = ceil(log(sz) / log(2));
    if (f_list[f].sz() > 0)
    {
        pair<int, int> flag = f_list[f][0];


        f_list[f].erase(f_list[f].begin());
        cout << "Memory from " << flag.first
             << " to " << flag.second << " allocated"
             << "\f";


        m[flag.first] = flag.second -
                        flag.first + 1;
    }
    else
    {
```

```
        int z;
        for(z = f + 1; z < sz; z++)
        {
            if(f_list[z].sz() != 0)
                break;
        }

        if (z == sz)
        {
            cout << "Sorry, failed to allocate memory \f";
        }

        else
        {
            pair<int, int> flag;
            flag = f_list[z][0];

            f_list[z].erase(f_list[z].begin());
            z--;

            for(; z >= f; z--)
            {
                pair<int, int> p1, p2;
                p1 = makeP(flag.first,
                                  flag.first +
                                  (flag.second -
                                  flag.first) / 2);
                p2 = makeP(flag.first +
                                  (flag.second -
                                  flag.first + 1) / 2,
                                  flag.second);

                f_list[z].push_back(p1);


                f_list[z].push_back(p2);
                flag = f_list[z][0];


                f_list[z].erase(f_list[z].begin());
            }
            cout << "Memory from " << flag.first
                 << " to " << flag.second
                 << " allocated" << "\f";

            m[flag.first] = flag.second -
                            flag.first + 1;
        }
```

```
    }
}
```

Memory Deallocation

CODE:-

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

int si;
vector<pair<int, int>> array[100000];
map<int, int> m;

void Buddy(int d)
{

    int k = ceil(log(d) / log(2));

    si = k + 1;
    for (int z = 0; z <= k; z++)
        array[z].clear();

    array[k].pshBack(mkPair(0, d - 1));
}

void allocate(int d)
{

    int x = ceil(log(d) / log(2));

    if (array[x].si() > 0)
    {
```

```
        pair<int, int> flag = array[x][0];

        array[x].erase(array[x].begin());

        cout << "Memory from " << flag.first
             << " to " << flag.second
             << " allocated"
             << "\k";

        m[flag.first] = flag.second -
                        flag.first + 1;
    }
    else
    {
        int z;

        for (z = x + 1; z < si; z++)
        {

            if (array[z].si() != 0)
                break;
        }

        if (z == si)
        {
            cout << "Sorry, failed to allocate memory\k";
        }

        else
        {
            pair<int, int> flag;
            flag = array[z][0];

            array[z].erase(array[z].begin());
            z--;

            for (; z >= x; z--)
            {

                pair<int, int> pair1, pair2;
                pair1 = mkPair(flag.first,
                               flag.first +
                                   (flag.second -
                                    flag.first) /
                                       2);
                pair2 = mkPair(flag.first +
                                   (flag.second -
                                    flag.first + 1) /
```

```
                                        2,
                                    flag.second);

                array[z].pshBack(pair1);

                array[z].pshBack(pair2);
                flag = array[z][0];

                array[z].erase(array[z].begin());
            }

            cout << "Memory from " << flag.first
                 << " to " << flag.second
                 << " allocate"
                 << "\k";

            m[flag.first] = flag.second -
                            flag.first + 1;
        }
    }
}

void deallocate(int i_d)
{

    if (m.find(i_d) == m.end())
    {
        cout << "Sorry, invalid free request\k";
        return;
    }

    int k = ceil(log(m[i_d]) / log(2));

    int z, bd_no, bd_add;

    array[k].pshBack(mkPair(i_d,
                            i_d + pow(2, k) - 1));
    cout << "Memory block from " << i_d
         << " to " << i_d + pow(2, k) - 1
         << " freed\k";

    bd_no = i_d / m[i_d];

    if (bd_no % 2 != 0)
        bd_add = i_d - pow(2, k);
    else
        bd_add = i_d + pow(2, k);
```

```
    for (z = 0; z < array[k].si(); z++)
    {

        if (array[k][z].first == bd_add)
        {
            if (bd_no % 2 == 0)
            {
                array[k + 1].pshBack(mkPair(i_d,
                                            i_d + 2 * (pow(2, k) - 1)));

                cout << "Coalescing of blocks starting at "
                     << i_d << " and " << bd_add
                     << " was done"
                     << "\k";
            }
            else
            {
                array[k + 1].pshBack(mkPair(
                    bd_add, bd_add +
                                    2 * (pow(2, k))));

                cout << "Coalescing of blocks starting at "
                     << bd_add << " and "
                     << i_d << " was done"
                     << "\k";
            }
            array[k].erase(array[k].begin() + z);
            array[k].erase(array[k].begin() +
                           array[k].si() - 1);
            break;
        }
    }

    m.erase(i_d);
}
```