End semester OS lab exam

AU1940127

Jay Ghetia

Question 1

Question 2

Allocation

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

int size;

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

map<int, int> mp;

void initialize(int sz)
{

    int n = ceil(log(sz) / log(2));
    size = n + 1;

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


    free_list[n].push_back(make_pair(0, sz - 1));
}

void allocate(int sz)
{

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


    if (free_list[n].size() > 0)
    {
        pair<int, int> temp = free_list[n][0];


        free_list[n].erase(free_list[n].begin());
        cout << "Memory from " << temp.first
            << " to " << temp.second << " allocated"
            << "\n";
        mp[temp.first] = temp.second -
                        temp.first + 1;
    }
    else
    {
        int i;
        for(i = n + 1; i < size; i++)
        {


            if(free_list[i].size() != 0)
                break;
        }

        if (i == size)
        {
            cout << "Sorry, failed to allocate memory \n";
        }


        else
        {
            pair<int, int> temp;
            temp = free_list[i][0];

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

            for(; i >= n; 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);

                free_list[i].push_back(pair1);


                free_list[i].push_back(pair2);
                temp = free_list[i][0];
                free_list[i].erase(free_list[i].begin());
            }
            cout << "Memory from " << temp.first
                << " to " << temp.second
                << " allocated" << "\n";

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


int main()
{



    initialize(128);
    allocate(32);
    allocate(7);
    allocate(64);
    allocate(56);

    return 0;
}
```

Deallocation

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


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


map<int, int> mp;

void Buddy(int s)
{

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

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

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

void allocate(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" << "\n";

        mp[temp.first] = temp.second -
                        temp.first + 1;
    }
    else
    {
        int i;

        for(i = x + 1; i < size; i++)
        {

            if (arr[i].size() != 0)
                break;
        }


        if (i == size)
        {
            cout << "Sorry, failed to allocate memory\n";
        }

        else
        {
            pair<int, int> temp;
            temp = arr[i][0];


            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(int id)
{


    if(mp.find(id) == mp.end())
    {
        cout << "Sorry, invalid free request\n";
        return;
    }


    int n = ceil(log(mp[id]) / log(2));

    int i, buddyNumber, buddyAddress;


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


    buddyNumber = id / mp[id];

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

    for(i = 0; i < arr[n].size(); i++)
    {


        if (arr[n][i].first == buddyAddress)
        {

            if (buddyNumber % 2 == 0)
            {
                arr[n + 1].push_back(make_pair(id,
                id + 2 * (pow(2, n) - 1)));

                cout << "Coalescing of blocks starting at "
                    << id << " and " << buddyAddress
                    << " was done" << "\n";
            }
            else
            {
                arr[n + 1].push_back(make_pair(
                    buddyAddress, buddyAddress +
                    2 * (pow(2, n))));

                cout << "Coalescing of blocks starting at "
                    << buddyAddress << " 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(128);
    allocate(16);
    allocate(16);
    allocate(16);
    allocate(16);
    deallocate(0);
    deallocate(9);
    deallocate(32);
    deallocate(16);

    return 0;
}
```

Question 3

```
#include <pthread.h>
#include <semaphore.h>
#include <stdlib.h>
#include <stdio.h>
#define MaxItems 5
#define BufferSize 5
sem_t empty;
sem_t full;
int in = 0;
int out = 0;
int buffer[BufferSize];
pthread_mutex_t mutex;
void *producer(void *pno)
{
      int item;
       for(int i = 0; i < MaxItems; i++) {
      item = rand(); // Produce an random item
      sem_wait(&empty);
      pthread_mutex_lock(&mutex);
      buffer[in] = item;
      printf("Producer %d: Insert Item %d at %d\n", *((int
*)pno),buffer[in],in);

      in = (in+1)%BufferSize;
      pthread_mutex_unlock(&mutex);
      sem_post(&full);
      }
}
void *consumer(void *cno)
{
 for(int i = 0; i < MaxItems; i++)
      {
       sem_wait(&full);
       pthread_mutex_lock(&mutex);
       int item = buffer[out];
      printf("Consumer %d: Remove Item %d from %d\n",*((int *)cno),item,
out);
       out = (out+1)%BufferSize;

       pthread_mutex_unlock(&mutex);
       sem_post(&empty);
       }
}
int main()
{
      pthread_t pro[5],con[5];
      pthread_mutex_init(&mutex, NULL);
      sem_init(&empty,0,BufferSize);
       sem_init(&full,0,0);
      int a[5] = {1,2,3,4,5};
      for(int i = 0; i < 5; i++)
       {
            pthread_create(&pro[i], NULL, (void *)producer, (void *)&a[i]);
       }
       for(int i = 0; i < 5; i++)
      {
            pthread_create(&con[i], NULL, (void *)consumer, (void *)&a[i]);
      }
       for(int i = 0; i < 5; i++) {
             pthread_join(pro[i], NULL);
       }
       for(int i = 0; i < 5; i++) {
             pthread_join(con[i], NULL);
      }
      pthread_mutex_destroy(&mutex);
      sem_destroy(&empty);
      sem_destroy(&full);
 return 0;

}
```