****

**CSE 332: Operating System**

**Section 2, Monsoon Semester 2021**

**End Sem Exam - Codes**

**Dhruv Prajapati**

**AU1940192**

**Q-1(b)**

#include<bits/stdc++.h>

#include<pthread.h>

#include<stdio.h>

#include<stdlib.h>

#include<semaphore.h>

sem\_t semaphoreD;

int temp1 = 0;

void \**threadFunction1*(void \*arg)

{

    int i;

    for(i=0;i<26;i++)

    {

*sem\_wait*(&semaphoreD);

        if(temp1%3 != 1) i--;

        else

        {

*printf*("%d ",1+i);

            temp1++;

        }

*sem\_post*(&semaphoreD);

    }

*pthread\_exit*(*NULL*);

}

void \**threadFunction2*(void \*arg)

{

    int i;

    for(i=0;i<26;i++)

    {

*sem\_wait*(&semaphoreD);

        if(temp1%3 != 0) i--;

        else

        {

*printf*("%c ",'A'+i);

            temp1++;

        }

*sem\_post*(&semaphoreD);

    }

*pthread\_exit*(*NULL*);

}

void \**threadfunction3*(void \*arg){

    int i;

    for(i=0;i<26;i++)

    {

*sem\_wait*(&semaphoreD);

        if(temp1%3 != 2) i--;

        else

        {

*printf*("%c ",'a'+i);

            temp1++;

        }

*sem\_post*(&semaphoreD);

    }

*pthread\_exit*(*NULL*);

}

int *main*()

{

    pthread\_t thr1, thr2, thr3;

    char stA = 'A';

    char sta = 'a';

*sem\_init*(&semaphoreD, 0, 1);

*pthread\_create*(&thr1,*NULL*,*threadFunction2*,*NULL*);

*pthread\_create*(&thr2,*NULL*,*threadFunction1*,*NULL*);

*pthread\_create*(&thr3,*NULL*,*threadfunction3*,*NULL*);

*pthread\_join*(thr1,*NULL*);

*pthread\_join*(thr2,*NULL*);

*pthread\_join*(thr3,*NULL*);

*sem\_destroy*(&semaphoreD);

    return 0;

}

**Q-2(b)**

**Memory-Allocation Program**

#include<bits/stdc++.h>

using namespace std;

int sz;

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

map<int, int> mp;

void *buddy1\_initialize*(int sz)

{

    int num = *ceil*(*log*(sz) / *log*(2));

    sz = num + 1;

    for(int i = 0; i <= num; i++)

        freeList[i].*clear*();

    freeList[num].*push\_back*(*make\_pair*(0, sz - 1));

}

void *allocate*(int sz)

{

    int num = *ceil*(*log*(sz) / *log*(2));

    if (freeList[num].*size*() > 0)

    {

        pair<int, int> tmp = freeList[num]*[*0*]*;

        freeList[num].*erase*(freeList[num].*begin*());

        cout*<<*"Memory from "*<<*tmp.first*<<*" to "*<<*tmp.second*<<*" allocated"

*<<* "\n";

        mp*[*tmp.first*]* = tmp.second - tmp.first + 1;

    }

    else

    {

        int i;

        for(i = num + 1; i < sz; i++)

        {

            if(freeList[i].*size*() != 0)

                break;

        }

        if (i == sz)

        {

            cout*<<*"Sorry, Memory is not allotted.\n";

        }

        else

        {

            pair<int, int> tmp;

            tmp *=* freeList[i]*[*0*]*;

            freeList[i].*erase*(freeList[i].*begin*());

            i--;

            for(;i>= num;i--)

            {

                pair<int, int> pair1, pair2;

                pair1 *=* *make\_pair*(tmp.first,tmp.first + (tmp.second - tmp.first) / 2);

                pair2 *=* *make\_pair*(tmp.first + (tmp.second - tmp.first + 1) / 2, tmp.second);

                freeList[i].*push\_back*(pair1);

                freeList[i].*push\_back*(pair2);

                tmp *=* freeList[i]*[*0*]*;

                freeList[i].*erase*(freeList[i].*begin*());

            }

            cout*<<*"Memory from "*<<*tmp.first*<<*" to "*<<*tmp.second*<<* " is allocated\n";

            mp*[*tmp.first*]* = tmp.second - tmp.first + 1;

        }

    }

}

int *main*()

{

*//driver code*

*buddy1\_initialize*(256);

*allocate*(4);

*allocate*(8);

*allocate*(16);

*allocate*(32);

    return 0;

}

**Memory Deallocation Program**

#include<bits/stdc++.h>

using namespace std;

int sz;

vector<pair<int, int>> arr1[100000];

map<int, int> mp;

void *BuddyFunction*(int s)

{

    int num = *ceil*(*log*(s) / *log*(2));

    sz = num + 1;

    for(int i = 0; i <= num; i++)

        arr1[i].*clear*();

    arr1[num].*push\_back*(*make\_pair*(0, s - 1));

}

void *allocate*(int s)

{

    int cnt = *ceil*(*log*(s) / *log*(2));

    if (arr1[cnt].*size*() > 0)

    {

        pair<int, int> tmp = arr1[cnt]*[*0*]*;

        arr1[cnt].*erase*(arr1[cnt].*begin*());

        cout*<<*"Memory from "*<<*tmp.first*<<* " to " *<<* tmp.second*<<* " is allocated.\n";

        mp*[*tmp.first*]* = tmp.second - tmp.first + 1;

    }

    else

    {

        int i;

        for(i = cnt + 1; i < sz; i++)

        {

            if (arr1[i].*size*() != 0) break;

        }

        if (i == sz)

        {

            cout *<<* "Sorry, Memory is not aloocated.\n";

        }

        else

        {

            pair<int, int> tmp;

            tmp *=* arr1[i]*[*0*]*;

            arr1[i].*erase*(arr1[i].*begin*());

            i--;

            for(;i >= cnt; i--)

            {

                pair<int, int> pair1, pair2;

                pair1 *=* *make\_pair*(tmp.first,

                                tmp.first +

                                (tmp.second -

                                tmp.first) / 2);

                pair2 *=* *make\_pair*(tmp.first +

                                (tmp.second -

                                tmp.first + 1) / 2,

                                tmp.second);

                arr1[i].*push\_back*(pair1);

                arr1[i].*push\_back*(pair2);

                tmp *=* arr1[i]*[*0*]*;

                arr1[i].*erase*(arr1[i].*begin*());

            }

            cout*<<*"Memory from "*<<*tmp.first*<<* " to " *<<* tmp.second*<<* " is allocated.\n";

            mp*[*tmp.first*]* = tmp.second -

                            tmp.first + 1;

        }

    }

}

void *deallocate*(int id)

{

    if(mp.*find*(id) *==* mp.*end*())

    {

        cout *<<* "Invalid free request! Please Try Again\n";

        return;

    }

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

    int i, number, address;

    arr1[n].*push\_back*(*make\_pair*(id,

                            id + *pow*(2, n) - 1));

    cout *<<* "Memory block from " *<<* id

*<<* " to "*<<* id + *pow*(2, n) - 1

*<<* " freed\n";

    number = id / mp*[*id*]*;

    if (number % 2 != 0) address = id - *pow*(2, n);

    else address = id + *pow*(2, n);

    for(i = 0; i < arr1[n].*size*(); i++)

    {

        if (arr1[n]*[*i*]*.first == address)

        {

            if (number % 2 == 0)

            {

                arr1[n + 1].*push\_back*(*make\_pair*(id,

                id + 2 \* (*pow*(2, n) - 1)));

                cout *<<* "Coalescing of blocks starting at "*<<* id *<<* " and " *<<* address*<<* " was done./n";

            }

            else

            {

                arr1[n + 1].*push\_back*(*make\_pair*(

                    address, address +

                    2 \* (*pow*(2, n))));

                cout *<<* "Coalescing of blocks starting at "*<<* id *<<* " and " *<<* address*<<* " was done./n";

            }

            arr1[n].*erase*(arr1[n].*begin*() *+* i);

            arr1[n].*erase*(arr1[n].*begin*() *+*

            arr1[n].*size*() *-* 1);

            break;

        }

    }

    mp.*erase*(id);

}

int *main*()

{

//Driver code

*BuddyFunctionFunction*(256);

*allocate*(32);

*allocate*(32);

*allocate*(32);

*allocate*(32);

*deallocate*(0);

*deallocate*(18);

*deallocate*(64);

*deallocate*(32);

    return 0;

}