**AU1940313 Pranav Gandhi**

**End Sem Os practical exam**

**Ans1)**

**Multithreads:-**

**Code:-**

#*include*<stdio.h>

#*include*<pthread.h>

#*include*<stdlib.h>

#*include*<semaphore.h>

sem\_t semph; //*semaphore*

int l = 0; //*lock variable*

void \**thread*(void \*arg){

int j;

*for*(j=0;j<26;j++){

*sem\_wait*(&semph);

*if*(l%3 != 1){

j--;

}

*else*{

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

l++;

}

*sem\_post*(&semph);

}

*pthread\_exit*(*NULL*);

}

void \**thread\_2*(void \*arg){

int a;

*for*(a=0;a<26;a++){

*sem\_wait*(&semph);

*if*(l%3 != 0){

a--;

}

*else*{

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

l++;

}

*sem\_post*(&semph);

}

*pthread\_exit*(*NULL*);

}

void \**thread\_3*(void \*arg){

int i;

*for*(i=0;i<26;i++){

*sem\_wait*(&semph);

*if*(l%3 != 2){

i--;

}

*else*{

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

l++;

}

*sem\_post*(&semph);

}

*pthread\_exit*(*NULL*);

}

int *main*(){

pthread\_t t1, t2, t3; //*threads*

char start1 = 'A';

char start2 = 'a';

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

*pthread\_create*(&t1,*NULL*,*thread*,*NULL*);

*pthread\_create*(&t2,*NULL*,*thread\_2*,*NULL*);

*pthread\_create*(&t3,*NULL*,*thread\_3*,*NULL*);

*pthread\_join*(t1,*NULL*);

*pthread\_join*(t2,*NULL*);

*pthread\_join*(t3,*NULL*);

*sem\_destroy*(&semph);

*return* 0;

}

**Ans2) Buddy’s Algorithm:-**

**Code of allocate:-**

**#*include*<bits/stdc++.h>**

**using namespace std;**

**int s;//*size***

**vector<pair<int, int>> free\_list[100000];**

**map<int, int> mp;**

**void *initialize*(int x)**

**{**

**int n = *ceil*(*log*(x) / *log*(2));**

**s = n + 1;**

***for*(int i = 0; i <= n; i++)**

**free\_list[i].*clear*();**

**free\_list[n].*push\_back*(*make\_pair*(0, x - 1));**

**}**

**void *allocate*(int x)**

**{**

**int n = *ceil*(*log*(x) / *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 j;**

***for*(j = n + 1; j < s; i++)**

**{**

***if*(free\_list[j].*size*() != 0)**

***break*;**

**}**

***if* (j == s)**

**{**

**cout *<<* "Sorry, failed to allocate memory \n";**

**}**

***else***

**{**

**pair<int, int> temp;**

**temp *=* free\_list[j]*[*0*]*;**

**free\_list[j].*erase*(free\_list[j].*begin*());**

**j--;**

***for*(; j >= n; j--)**

**{**

**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[j].*push\_back*(pair1);**

**// *Push them in free list***

**free\_list[j].*push\_back*(pair2);**

**temp *=* free\_list[j]*[*0*]*;**

**free\_list[j].*erase*(free\_list[j].*begin*());**

**}**

**cout *<<* "Memory from " *<<* temp.first**

***<<* " to " *<<* temp.second**

***<<* " allocated" *<<* "\n";**

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

**temp.first + 1;**

**}**

**}**

**}**

**int *main*()**

**{**

***initialize*(512);**

***allocate*(33);**

***allocate*(10);**

***allocate*(14);**

***allocate*(24);**

***return* 0;**

**}**

**Deallocation Code:-**

**#*include*<bits/stdc++.h>**

**using namespace std;**

**int sz;**

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

**map<int, int> mp;**

**void *Buddy*(int x)**

**{**

**int n = *ceil*(*log*(x) / *log*(2));**

**sz = n + 1;**

***for*(int i = 0; i <= n; i++)**

**arr[i].*clear*();**

**arr[n].*push\_back*(*make\_pair*(0, - 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 < sz; i++)**

**{**

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

***break*;**

**}**

***if* (i == sz)**

**{**

**cout *<<* "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 frm " *<<* 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;**

**}**