Ahmedabad University
OS Lab Codes

Enrollment no: AU1940207 Name: Keyur Nagar

Answer 1:

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



int i=0;
void* routine1(){
   int capital=i+65;
   printf("%c ",capital);
}
void* routine2(){
   int number=i+1;
   printf("%d ",number);
}
void* routine3(){
   int small=i+97;
   printf("%c ",small);

}

int main(int args, char* argv[]){
   pthread_t t1,t2,t3;


   for(i=0;i<26;i++){
      pthread_create(&t1, NULL,&routine1,NULL);
      pthread_join(t1, NULL);
      pthread_create(&t2, NULL,&routine2,NULL);
      pthread_join(t2, NULL);
      pthread_create(&t3, NULL,&routine3,NULL);
      pthread_join(t3, NULL);
   }


   return 0;
}
```

Answer 2B:

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


int size;


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


map<int, int> mp;

void initialize(int sz)
{

	//matemoimum number of partition possible
	int n = ceil(log(sz) / log(2));

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


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

void allocate(int s)
{

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

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

		memory[temo].erase(memory[temo].begin());
```

```
            cout << "Memory address starting from " << temp.first
                 << " to " << temp.second
                 << " has been allocated" << "\n";


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


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


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


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


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


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

                for(;i >= temo; 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);

                        memory[i].push_back(pair1);


                        memory[i].push_back(pair2);
                        temp = memory[i][0];


                        memory[i].erase(memory[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, startingaddress;


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

```
        buddyNumber = id / mp[id];

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


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


                if (memory[n][i].first == startingaddress)
                {


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

                                cout << "merging of blocks starting at " << id << " and " <<
startingaddress << " has been completed" << "\n";
                        }
                        else
                        {
                                memory[n + 1].push_back(make_pair(startingaddress,
startingaddress + 2 * (pow(2, n))));

                                cout << "merging of blocks starting at " << startingaddress
<< " and " << id << " has been completed" << "\n";
                        }
                        memory[n].erase(memory[n].begin() + i);
                        memory[n].erase(memory[n].begin() +
                        memory[n].size() - 1);
                        break;
                }
        }


        mp.erase(id);
}

// Driver code
int main()
```

```
{

        initialize(128);
        allocate(16);
        allocate(16);
        allocate(16);
        allocate(16);
        deallocate(16);
        deallocate(0);
        deallocate(9);
        deallocate(32);


        return 0;
}
```