CS266 ASSIGNMENT 4

NAME:

ARCHIT AGRAWAL

ROLL NO. :

202051213

SECTION:

2

Problem

Write a Multi-Threaded program that can take your fullname(FName MName LName) and roll

number as input and simultaneously perform the following operations.

– Reverse of the string

– Print all permutations of the first four characters of the name with repetition.

– Rearrange your first name so that all the same characters become d distance apart

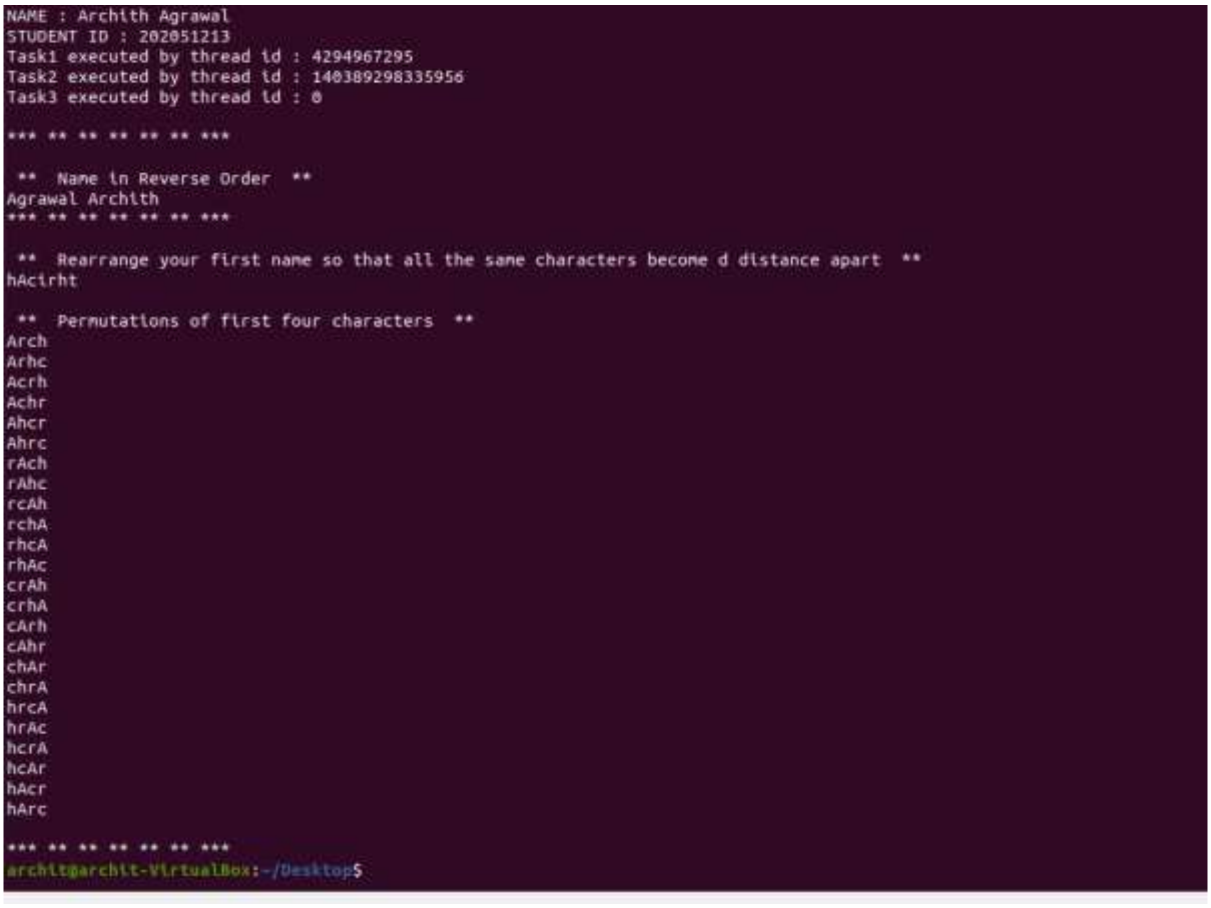
Code

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>
#include<string.h>
#define MAX 256

int d;

void *reverse(char str[]){
    int i, j, len, startIndex, endIndex;
    len = strlen(str);
    endIndex = len - 1;

    printf("\n **  Name in Reverse Order  ** \n");
    for(i = len - 1; i >= 0; i--){
        if(str[i] == ' ' || i == 0){
            if(i == 0){
                startIndex = 0;
            }
            else{
                startIndex = i + 1;
            }
            for(j = startIndex; j <= endIndex; j++){
                printf("%c", str[j]);
            }
            endIndex = i - 1;
            printf(" ");
        }
    }
    printf("\n*** ** ** ** ** ** ***\n");
}

/* Function to swap values at
   two pointers */
void swap(char *x, char *y) {
    char temp;
    temp = *x;
    *x = *y;
    *y = temp;
}

/* Function to print permutations
   of string
   This function takes three parameters:
   1. String
   2. Starting index of the string
   3. Ending index of the string. */
void permute(char *a, int l, int r) {
    int i;
    if (l == r) printf("%s\n", a);
    else{
        for (i = l; i <= r; i++){
            swap((a + l), (a + i));
            permute(a, l + 1, r);

            //backtrack
            swap((a + l), (a + i));
        }
    }
}

void *permutation(char *str){
    printf("\n **  Permutations of first four characters  ** \n");
    permute(str, 0, 3);
    printf("\n*** ** ** ** ** ** ***\n");
}

// A structure to store a character 'c' and its frequency
// 'f' in input string
typedef struct charFreq {
    char c;
    int f;
} charFreq ;

// A utility function to swap two charFreq items.
void swapFreq(charFreq* x, charFreq* y){
    charFreq z = *x;
    *x = *y;
    *y = z;
}

// A utility function to maxheapify the node freq[i] of a
// heap stored in freq[]
void maxHeapify(charFreq freq[], int i, int heap_size){
    int l = i * 2 + 1;
    int r = i * 2 + 2;
    int largest = i;
    if (l < heap_size && freq[l].f > freq[i].f) largest = l;
    if (r < heap_size && freq[r].f > freq[largest].f) largest = r;
    if (largest != i) {
        swapFreq(&freq[i], &freq[largest]);
        maxHeapify(freq, largest, heap_size);
    }
}

// A utility function to convert the array freq[] to a max
// heap
void buildHeap(charFreq freq[], int n){
    int i = (n - 1) / 2;
    while (i >= 0) {
        maxHeapify(freq, i, n);
        i--;
    }
}

// A utility function to remove the max item or root from
// max heap
charFreq extractMax(charFreq freq[], int heap_size){
    charFreq root = freq[0];
    if (heap_size > 1) {
        freq[0] = freq[heap_size - 1];
        maxHeapify(freq, 0, heap_size - 1);
    }
    return root;
}

void *rearrange(char str[]){

    // Find length of input string
    int n = strlen(str);

    // Create an array to store all characters and their
    // frequencies in str[]
    charFreq freq[MAX] = { { 0, 0 } };
```

```c
    int m = 0; // To store count of distinct characters in
               // str[]

    // Traverse the input string and store frequencies of
    // all characters in freq[] array.
    for (int i = 0; i < n; i++) {
        char x = str[i];

        // If this character has occurred first time,
        // increment m
        if (freq[x].c == 0)
            freq[x].c = x, m++;

        (freq[x].f)++;
        str[i] = '\0'; // This change is used later
    }

    // Build a max heap of all characters
    buildHeap(freq, MAX);

    // Now one by one extract all distinct characters from
    // max heap and put them back in str[] with the d
    // distance constraint
    for (int i = 0; i < m; i++) {
        charFreq x = extractMax(freq, MAX - i);

        // Find the first available position in str[]
        int p = i;
        while (str[p] != '\0')
            p++;

        // Fill x.c at p, p+d, p+2d, .. p+(f-1)d
        for (int k = 0; k < x.f; k++) {
            // If the index goes beyond size, then string
            // cannot be rearranged.
            if (p + d * k >= n) {
                printf("Cannot be rearranged");
                exit(0);
            }
            str[p + d * k] = x.c;
        }
    }

    printf("\n **  Rearrange your first name so that all the same characters
become d distance apart  ** \n");
    printf("%s\n",str);
}
```

```c
int main(){
    //char name[50];

    long int roll;

    char part[5];

    //Here it is expected that the person
    //does not start inputting his/her name with a space
    //fgets(name, 50, stdin);
    //scanf("%ld", &roll);


    char name[] = "Archith Agrawal";
    roll = 202051213;

    int i;
    char *ptr = strchr(name, ' ');
    if(ptr) {
       i = ptr - name;
    }
    else{
           i = strlen(name);
    }
    char firstname[i+1];

    for(int x =0; x < i; x++) firstname[x] = name[x];

    firstname[i] = '\0';

    strncpy(part,name,4);

    //in my name length of Firstname L =7

    d = 2+5+3;

    //since L < d
    d = 5;

    pthread_t t1;
    pthread_t t2;
    pthread_t t3;

    printf("NAME : %s\n",name);
    printf("STUDENT ID : %ld\n",roll);
    printf("Task1 executed by thread id : %ld\n", (long*)(void*)t1);
    printf("Task2 executed by thread id : %ld\n", (long*)(void*)t2);
    printf("Task3 executed by thread id : %ld\n", (long*)(void*)t3);
    pthread_create(&t1, NULL, reverse, name);
    //pthread_exit(NULL);
    pthread_create(&t2, NULL, permutation, part);
    //pthread_exit(NULL);
    pthread_create(&t3, NULL, rearrange, firstname);
    printf("\n*** ** ** ** ** ** ***\n");
    pthread_exit(NULL);
    return 0;
}
```

Output



```
NAME : Archith Agrawal
STUDENT ID : 202051213
Task1 executed by thread id : 4294967295
Task2 executed by thread id : 140389298335956
Task3 executed by thread id : 0

*** ** ** ** ** ** ***

 **  Name in Reverse Order  **
Agrawal Archith
*** ** ** ** ** ** ***

 **  Rearrange your first name so that all the same characters become d distance apart  **
hAcirht

 **  Permutations of first four characters  **
Arch
Arhc
Acrh
Achr
Ahcr
Ahrc
rAch
rAhc
rcAh
rchA
rhcA
rhAc
crAh
crhA
cArh
cAhr
chAr
chrA
hrcA
hrAc
hcrA
hcAr
hAcr
hArc

*** ** ** ** ** ** ***
archit@archit-VirtualBox:~/Desktop$
```