Practical 2
Implementation of B+ Tree

Gahan M. Saraiya, 18MCEC10

18mcec10@nirmauni.ac.in

I. Introduction

Aim of this practical is to implement C program to calculate exponential value for number using divide and conquer.

II. Implementation

I. Utility *utility.h*

```
//
// Created by jarvis on 17/8/18.
//

#ifndef DSA_LAB_UTILITY_H
#define DSA_LAB_UTILITY_H

#include <string.h>
#include <stdarg.h>

int write_log(const char *format, ...) {
    if(DEBUG) {
        va_list args;
        va_start (args, format);
        vprintf(format, args);
        va_end (args);
    }
}

int *get_min_max(int *array, int no_of_elements, int min_max[]){
    // get minimum and maximum of array
//    printf("elements of array: ");
    for(int i=0; i<no_of_elements; i++){
//        printf("%d ", *(array + i));
        if (*(array + i) < min_max[0])
            min_max[0] = *(array + i);
        if (*(array + i) > min_max[1])
            min_max[1] = *(array + i);
    }
```

```c
    return min_max;
}

int display_array(int *array, int no_of_elements){
    // display given array of given size(no. of elements require because sizeof()
     ↪  returns max bound value)
    write_log(": ");
    for(int i=0; i<no_of_elements; i++){
        write_log( "%d ", *(array + i));
    }
    return 0;
}


void swap(int *one, int *two){
    // swap function to swap elements by location/address
    int temp = *one;
    *one = *two;
    *two = temp;
}


void read_file_input() {
    // under development function to read inputs from file
    int ptr[100], count = 0, i, ar_count;
    char c[100];
    FILE *fp = fopen("file.in", "r");

    char in = fgetc(fp);
    // ar_count = (int) (in - '0');
    printf("\narr\n");
    while (in != EOF){
        if ((int) (in -'0') == -16){
            printf("\nspace\n");
        }
        else{
            printf("%c - %d\n",in,  (int) (in - '0'));
        }
        in = fgetc(fp);
    }
    printf("\n\n");
    fclose (fp);
}

#endif //DSA_LAB_UTILITY_H
```

II. Main Program - *recursive_exponential.c*

```c
//
// Created by Gahan Saraiya on 1/10/18.
// Recursive Exponential algorithm using Divide and Conquer Approach
//
#include <stdio.h>
#include <stdlib.h>

int exponent(int number, int power){
    int new_power;
    // terminating condition
    if (power == 0) {
        return 1;
    }
    if (power == 1){
        return number;
    }
    else if (power == 2){
        return number * number;
    }
    else if (power > 2){
        new_power = power / 2;
        int sub_result;
        sub_result = exponent(number * number, new_power);
        // recursive exponential
        if (power % 2 == 0) {
            return sub_result;
        } else {
            return number * sub_result;
        }
    }
}

int main(int argc, char *argv[]){
    int power, result, number;
    printf("Enter Number: ");
    scanf("%d", &number);
    printf("Enter Power: ");
    scanf("%d", &power);

    // recursion call
    result = exponent(number, power);
    printf("Answer for %d ^ %d : %d\n", number, power, result);
}
```

II.1 Output

```
Enter Number: 15
Enter Power: 7
Answer for 15 ^ 7 : 170859375
```

II.2 Output

```
Enter Number: 5
Enter Power: 30
Answer for 5 ^ 30 : 433305513
```