Practical 5 B
Implementation of Recursive Exponential algorithm using Divide and Conquer Approach

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 max(int a, int b) { return (a > b)? a : b; }
int min(int a, int b) { return (a < b)? a : b; }

int write_log(const char *format, ...) {
    if(DEBUG) {
        printf("\n[DEBUG_LOG]> ");
        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
```

```
26  //    printf("elements of array: ");
27      for(int i=0; i<no_of_elements; i++){
28  //        printf("%d ", *(array + i));
29          if (*(array + i) < min_max[0])
30              min_max[0] = *(array + i);
31          if (*(array + i) > min_max[1])
32              min_max[1] = *(array + i);
33      }
34      return min_max;
35  }
36
37  int display_array(int *array, int no_of_elements){
38      // display given array of given size(no. of elements require because sizeof()
        ↪  returns max bound value)
39      write_log(": ");
40      for(int i=0; i<no_of_elements; i++){
41          write_log( "%d ", *(array + i));
42      }
43      return 0;
44  }
45
46  int show_2d_array(int **array, int no_of_elements){
47      // display given array of given size(no. of elements require because sizeof()
        ↪  returns max bound value)
48      write_log(": ");
49      for(int i=0; i<no_of_elements; i++){
50          printf("a[%d][i]: ", i);
51          for(int j=0; j<no_of_elements; j++) {
52  //            printf("array[%d][%d]: %d ", i, j, array[i][j]);
53              printf("%d\t", array[i][j]);
54          }
55          printf("\twhere 0<=i<=%d\n", no_of_elements-1);
56      }
57      return 0;
58  }
59
60  int display_2d_array(int **array, int no_of_elements){
61      // display given array of given size(no. of elements require because sizeof()
        ↪  returns max bound value)
62      write_log(": ");
63      for(int i=0; i<no_of_elements; i++){
64          printf("a[%d][]: ", i);
65          for(int j=0; j<no_of_elements; j++) {
66  //            printf("array[%d][%d]: %d ", i, j, array[i][j]);
67              printf("%d ", array[i][j]);
68          }
69          printf("\n");
70      }
```

```
    return 0;
}


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

#endif //DSA_LAB_UTILITY_H
```

II. Main Program - *recursive_exponential.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;
        }
    }
}

```

```c
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
```