LAB ASSIGNMENT – 6

DISTRIBUTED SYSTEMS

NAME : SOURABH PATEL

ADMISSION NO. : U19CS082

Que : Simulate RPC (Create any one procedure on remote machine and call it from local machine)

List of programs for RPC

1. String is palindrome or not.
2. Find out if a given year is a Lear Year or not.
3. Find out the GCD of a given number.
4. Find out the Square root of a given number.
5. Swap two variables without using the 3rd variable.
6. Calculate Maximum, Minimum, average of given array.
7. Compare the given two strings.
8. Find out whether a given string is substring or not.
9. Concatenate the two strings.
10. Reverse the elements of an array.

All code in one file…. Sol.x

- ✓ sol.x

```
struct input{   int
size;   char str[50];
};

struct number{
  int a;   int b; };

struct num_arr{
  int size;   int
arr[50];
};

struct stat{   int
maximum;   int
minimum;   double
avg;
};

struct two_str{   struct input
s1;   struct input s2;
};

program SOL_PROG{
version SOL_VERS{     int
palindrome(input)=1;     int
check_leap_year(int)=2;
int gcd(number)=3;
double sqr_root(int)=4;
number swap(number)=5;
stat getstat(num_arr)=6;
int comp_str(two_str)=7;
int chk_substr(two_str)=8;
input concate(two_str)=9;
num_arr
reverse(num_arr)=10;
```

}=1;

}= 0x123456;

✓ sol_client.c

```c
/*
 * This is sample code generated by rpcgen.
 * These are only templates and you can use them
 * as a guideline for developing your own functions.
 */

#include "sol.h"
#include "string.h"

void
sol_prog_1(char *host)
{
    CLIENT *clnt;
    int  *result_1;
    input  palindrome_1_arg;
    int  *result_2;
    int  check_leap_year_1_arg;
    int  *result_3;
    number  gcd_1_arg;
    double  *result_4;
    int  sqr_root_1_arg;
    number  *result_5;
```

```
    number  swap_1_arg;
    stat  *result_6;
    num_arr  getstat_1_arg;
    int  *result_7;
    two_str  comp_str_1_arg;
    int  *result_8;
    two_str  chk_substr_1_arg;
    input  *result_9;
    two_str  concate_1_arg;
    num_arr  *result_10;
    num_arr  reverse_1_arg;

#ifndef DEBUG
    clnt = clnt_create (host, SOL_PROG, SOL_VERS, "udp");
    if (clnt == NULL) {
        clnt_pcreateerror (host);
        exit (1);
    }
#endif  /* DEBUG */

    // Main code starts from here
    int choice;

    printf("\n1. String is palindrome or not.");
    printf("\n2. Find out if a given year is a Lear Year or not.");
    printf("\n3. Find out the GCD of a given number.");
    printf("\n4. Find out the Square root of a given number.");
    printf("\n5. Swap two variables without using the 3rd variable.");
    printf("\n6. Calculate Maximum, Minimum, average of given array.");
    printf("\n7. Compare the given two strings.");
    printf("\n8. Find out whether a given string is substring or not.");
    printf("\n9. Concatenate the two strings.");
    printf("\n10. Reverse the elements of an array.");
    printf("\n11. Exit.");

    printf("\n\nEnter your choice: ");
    scanf("%d",&choice);
    int i;

    switch (choice) {
        case 1:
                printf("\nEnter the string: ");
                scanf("%s",palindrome_1_arg.str);
                palindrome_1_arg.size=strlen(palindrome_1_arg.str);
                //printf("%d\n",palindrome_1_arg.size);
                result_1 = palindrome_1(&palindrome_1_arg, clnt);
                if (result_1 == (int *) NULL) {
                    clnt_perror (clnt, "call failed");
```

```
                }
                else{
                    if(*result_1==1)
                        printf("\nYes, %s is palindrome.\n",palindrome_1_arg.str);
                    else
                        printf("\nNo, %s is not a
palindrome.\n",palindrome_1_arg.str);
                }
                break;
        case 2:
                printf("\nEnter the year: ");
                scanf("%d",&check_leap_year_1_arg);
                result_2 = check_leap_year_1(&check_leap_year_1_arg, clnt);
                printf("%d",*result_2);
                if (result_2 == (int *) NULL) {
                    clnt_perror (clnt, "call failed");
                }
                else{
                    if(*result_2==1)
                        printf("\nYes, %d is a leap year.\n",check_leap_year_1_arg);
                    else
                        printf("\nNo, %d is not a leap
year.\n",check_leap_year_1_arg);
                }
                break;
        case 3:
                printf("\nEnter the numbers: ");
                scanf("%d %d",&gcd_1_arg.a,&gcd_1_arg.b);
                result_3 = gcd_1(&gcd_1_arg, clnt);
                if (result_3 == (int *) NULL) {
                    clnt_perror (clnt, "call failed");
                }
                else{
                    printf("GCD(%d, %d) = %d\n",gcd_1_arg.a,gcd_1_arg.b,*result_3 );
                }
                break;
        case 4:
                printf("\nEnter the number: ");
                scanf("%d",&sqr_root_1_arg);
                result_4 = sqr_root_1(&sqr_root_1_arg, clnt);
                if (result_4 == (double *) NULL) {
                    clnt_perror (clnt, "call failed");
                }
                else{
                    printf("Square root of %d = %lf\n",sqr_root_1_arg,*result_4);
                }
                break;
        case 5:
```

```
                printf("\nEnter the numbers: ");
                scanf("%d %d",&swap_1_arg.a,&swap_1_arg.b);
                result_5 = swap_1(&swap_1_arg, clnt);
                if (result_5 == (number *) NULL) {
                    clnt_perror (clnt, "call failed");
                }
                else{
                    printf("Numbers after swapping (%d, %d)\n",result_5->a,result_5-
>b );
                }
                break;
        case 6:
                printf("\nEnter the size of array: ");
                scanf("%d",&getstat_1_arg.size);
                printf("\nEnter the elements of array: ");
                for(i=0;i<getstat_1_arg.size;i++){
                        scanf("%d",&getstat_1_arg.arr[i]);
                }
                result_6 = getstat_1(&getstat_1_arg, clnt);
                if (result_6 == (stat *) NULL) {
                    clnt_perror (clnt, "call failed");
                }
                else{
                    printf("Maximum element = %d\nMinimum element = %d\nAverage =
%lf \n",result_6->maximum,result_6->minimum,result_6->avg );
                }
                break;
        case 7:
                printf("\nEnter the first string: ");
                scanf("%s",comp_str_1_arg.s1.str);
                printf("\nEnter the second string: ");
                scanf("%s",comp_str_1_arg.s2.str);
                comp_str_1_arg.s1.size=strlen(comp_str_1_arg.s1.str);
                comp_str_1_arg.s2.size=strlen(comp_str_1_arg.s2.str);
                result_7 = comp_str_1(&comp_str_1_arg, clnt);
                if (result_7 == (int *) NULL) {
                    clnt_perror (clnt, "call failed");
                }
                else{
                    if(*result_7==0)
                        printf("Both strings are equal.\n");
                    else
                        printf("Both strings are not equal.\n");
                }
                break;
        case 8:
                printf("\nEnter the string: ");
                scanf("%s",chk_substr_1_arg.s1.str);
```

```
            printf("\nEnter the sub-string: ");
            scanf("%s",chk_substr_1_arg.s2.str);
            chk_substr_1_arg.s1.size=strlen(chk_substr_1_arg.s1.str);
            chk_substr_1_arg.s2.size=strlen(chk_substr_1_arg.s2.str);
            result_8 = chk_substr_1(&chk_substr_1_arg, clnt);
            if (result_8 == (int *) NULL) {
                clnt_perror (clnt, "call failed");
            }
            else{
                if(*result_8==0)
                    printf("Not a substring of given string.\n");
                else
                    printf("Yes, substring of given string.\n");
            }
            break;
        case 9:
            printf("\nEnter the first string: ");
            scanf("%s",concate_1_arg.s1.str);
            printf("\nEnter the second string: ");
            scanf("%s",concate_1_arg.s2.str);
            concate_1_arg.s1.size=strlen(concate_1_arg.s1.str);
            concate_1_arg.s2.size=strlen(concate_1_arg.s2.str);
            result_9 = concate_1(&concate_1_arg, clnt);
            if (result_9 == (input *) NULL) {
                clnt_perror (clnt, "call failed");
            }
            else{
                printf("Concatenated string = %s\n",result_9->str);
            }
            break;
        case 10:
            printf("\nEnter the size of array: ");
            scanf("%d",&reverse_1_arg.size);
            printf("\nEnter the elements of array: ");
            for(i=0;i<reverse_1_arg.size;i++){
                    scanf("%d",&reverse_1_arg.arr[i]);
            }
            result_10 = reverse_1(&reverse_1_arg, clnt);
            if (result_10 == (num_arr *) NULL) {
                clnt_perror (clnt, "call failed");
            }
            else{
                printf("\nReversed array: ");
                for(i=0;i<result_10->size;i++){
                    printf("%d ",result_10->arr[i] );
                }
                printf("\n");
            }
```

```
                break;
        case 11:
                printf("Thank you.\n");
                break;
        default:
                printf("Invalid input.\n");
                break;
    }

#ifndef DEBUG
    clnt_destroy (clnt);
#endif    /* DEBUG */
}


int
main (int argc, char *argv[])
{
    char *host;
    if (argc < 2) {
        printf ("usage: %s server_host\n", argv[0]);
        exit (1);
    }
    host = argv[1];
    sol_prog_1 (host);
exit (0);
}
```

sol_server.c

```
/*
 * This is sample code generated by rpcgen.
 * These are only templates and you can use them
 * as a guideline for developing your own functions.
 */

#include "sol.h"

int *
palindrome_1_svc(input *argp, struct svc_req *rqstp)
{
    static int  result=1;

    int n=argp[0].size;
    int i;
```

```
    double start=0,end=n,mid;
    while((end-start)>=0.000001){
        mid=(start+end)/2;
        if(mid*mid<n)
            start=mid;
        if(mid*mid>=n)
            end=mid;
    }
    result=mid;
    return &result;
}

number *
swap_1_svc(number *argp, struct svc_req *rqstp)
{
    static number  result;

    int a=argp[0].a;
    int b=argp[0].b;

    a=a+b;
    b=a-b;
    a=a-b;

    result.a=a;
    result.b=b;
    return &result;
}

stat *
getstat_1_svc(num_arr *argp, struct svc_req *rqstp)
{
    static stat  result;

    int n=argp[0].size;
    int max=-999999,min=999999;
    int i,sum=0;

    for(i=0;i<n;i++){
        if(argp[0].arr[i]>max)
            max=argp[0].arr[i];
        if(argp[0].arr[i]<min)
            min=argp[0].arr[i];
        sum+=argp[0].arr[i];
    }
    result.maximum=max;
    result.minimum=min;
    result.avg=(double)sum/n;
```

```
    for(i=0;i<n/2;i++){
        if(argp[0].str[i]!=argp[0].str[n-i-1]){
            result=0;
            break;
        }
    }
    return &result;
}

int *
check_leap_year_1_svc(int *argp, struct svc_req *rqstp)
{
    static int  result;
    int year=argp[0];

    if((year%4==0) && ((year%400==0) || (year%100!= 0)))
        result=1;
    else
        result=0;

    return &result;
}

int *
gcd_1_svc(number *argp, struct svc_req *rqstp)
{
    static int  result;
    int a = argp[0].a;
    int b = argp[0].b;

    int r;

    while (b > 0){
        r = a % b;
        a = b;
        b = r;
    }
    result=a;
    return &result;
}

double *
sqr_root_1_svc(int *argp, struct svc_req *rqstp)
{
    static double  result;

    int n=argp[0];
```

```
    return &result;
}

int *
comp_str_1_svc(two_str *argp, struct svc_req *rqstp)
{
    static int  result;

    result=strcmp(argp[0].s1.str,argp[0].s2.str);
    return &result;
}

int *
chk_substr_1_svc(two_str *argp, struct svc_req *rqstp)
{
    static int  result=0;

    if(strstr(argp[0].s1.str,argp[0].s2.str)!=NULL)
        result=1;

    return &result;
}

input *
concate_1_svc(two_str *argp, struct svc_req *rqstp)
{
    static input  result;

    strcpy(result.str,argp[0].s1.str);
    strcat(result.str,argp[0].s2.str);

    result.size=strlen(result.str);
    return &result;
}
num_arr *
reverse_1_svc(num_arr *argp, struct svc_req *rqstp)
{
    static num_arr  result;

    int n=argp[0].size;
    int i;

    for(i=0;i<n;i++){
        result.arr[i]=argp[0].arr[n-i-1];
    }
    result.size=n;
    return &result;
}
```

Output:

Server:

```
-VirtualBox:~/Downloads/U19CS022-DS-ASS-7$ ./sol_server
```

Client:

Palindrome:

```
1. String is palindrome or not.
2. Find out if a given year is a Lear Year or not.
3. Find out the GCD of a given number.
4. Find out the Square root of a given number.
5. Swap two variables without using the 3rd variable.
6. Calculate Maximum, Minimum, average of given array.
7. Compare the given two strings.
8. Find out whether a given string is substring or not.
9. Concatenate the two strings.
10. Reverse the elements of an array.
11. Exit.

Enter your choice: 1

Enter the string: abba

Yes, abba is palindrome.
```

Leap year:

```
1. String is palindrome or not.
2. Find out if a given year is a Lear Year or not.
3. Find out the GCD of a given number.
4. Find out the Square root of a given number.
5. Swap two variables without using the 3rd variable.
6. Calculate Maximum, Minimum, average of given array.
7. Compare the given two strings.
8. Find out whether a given string is substring or not.
9. Concatenate the two strings.
10. Reverse the elements of an array.
11. Exit.

Enter your choice: 2

Enter the year: 2016
1
Yes, 2016 is a leap year.
```

GCD:

```
Enter your choice: 3

Enter the numbers: 3
6
GCD(3, 6) = 3
```

Square:

```
Enter your choice: 4

Enter the number: 144
Square root of 144 = 12.000000
```

Swapping:

```
Enter your choice: 5

Enter the numbers: 8 9
Numbers after swapping (9, 8)
```

Avg:

```
1. String is palindrome or not.
2. Find out if a given year is a Lear Year or not.
3. Find out the GCD of a given number.
4. Find out the Square root of a given number.
5. Swap two variables without using the 3rd variable.
6. Calculate Maximum, Minimum, average of given array.
7. Compare the given two strings.
8. Find out whether a given string is substring or not.
9. Concatenate the two strings.
10. Reverse the elements of an array.
11. Exit.

Enter your choice: 6

Enter the size of array: 7

Enter the elements of array: 4 5 6 2 3 1 9
Maximum element = 9
Minimum element = 1
Average = 4.285714
```

String Compare:

```
Enter your choice: 7

Enter the first string: sandeep

Enter the second string: sundaram
Both strings are not equal.
```

IS substr:

```
Enter your choice: 8

Enter the string: sandeep

Enter the sub-string: and
Yes, substring of given string.
```

Concatination:

```
Enter your choice: 9

Enter the first string: sandeep

Enter the second string: rathod
Concatenated string = sandeeprathod
```

Reverse:

```
Enter your choice: 10

Enter the size of array: 5

Enter the elements of array: 1 2 3 4 5

Reversed array: 5 4 3 2 1
```