Distributed Systems (CS304)

Assignment - 6

U19CS012

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

List of Programs for RPC

1.) Find out the **Factorial of given Number**.

[factorial.x]

```
program factorial_PROG{
    version factorial_VERS{
        int factorial(int)=1;
    }=1;
}=0x4562877;
```

Run Command : `rpcgen –a –C factorial.x`

- ✓ All required files will be created.
- ✓ The factorial_client.c and factorial_server.c files would be modified as following:

[factorial_client.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 "factorial.h"

void
factorial_prog_1(char *host, int x)
{
    CLIENT *clnt;
    int  *result_1;
    int  factorial_1_arg;
```

```c
#ifndef DEBUG
    clnt = clnt_create (host, factorial_PROG, factorial_VERS, "udp");
    if (clnt == NULL) {

        clnt_pcreateerror (host);
        exit (1);
    }
#endif  /* DEBUG */
    factorial_1_arg = x;
    result_1 = factorial_1(&factorial_1_arg, clnt);
    if (result_1 == (int *) NULL) {
        clnt_perror (clnt, "call failed");
    }else{
        printf("Result : %d! = %d\n",x,*result_1);
    }
#ifndef DEBUG
    clnt_destroy (clnt);
#endif   /* DEBUG */
}

int
main (int argc, char *argv[])
{
    char *host;

    if (argc < 3) {
        printf ("usage: %s server_host\n", argv[0]);
        exit (1);
    }
    host = argv[1];
    factorial_prog_1 (host,atoi(argv[2]));
exit (0);
}
```

[factorial_server.c]

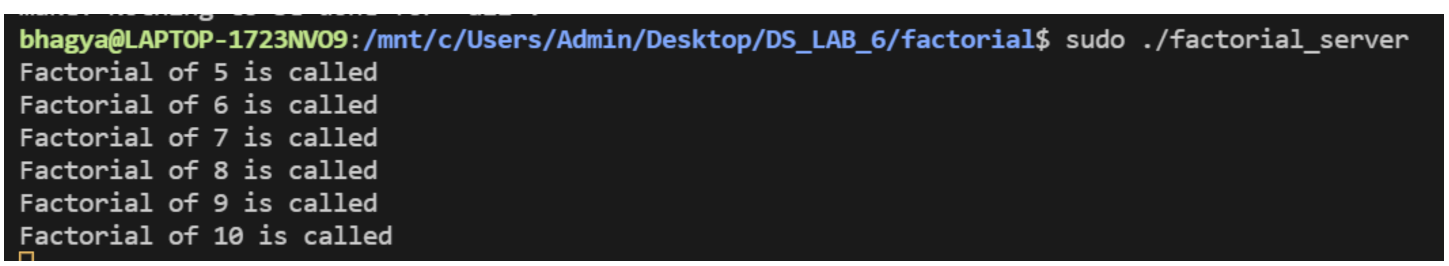
```c
/*
 * This is sample code generated by rpcgen.
 * These are only templates and you can use them
 * as a guideline for developing your own functions.
 */

#include "factorial.h"

int *
factorial_1_svc(int *argp, struct svc_req *rqstp)
{
    static int  result;
    /*
     * insert server code here
     */

    int  temp = 1;
    for(int i = 1; i <= *argp; i++)
    {
        temp *= i;
    }
    printf("Factorial of %d is called\n",*argp);
    result = temp;

    return &result;
}
```

Run Command : make -f Makefile.factorial

Very IMP Step - **Run Command** : sudo rpcbind

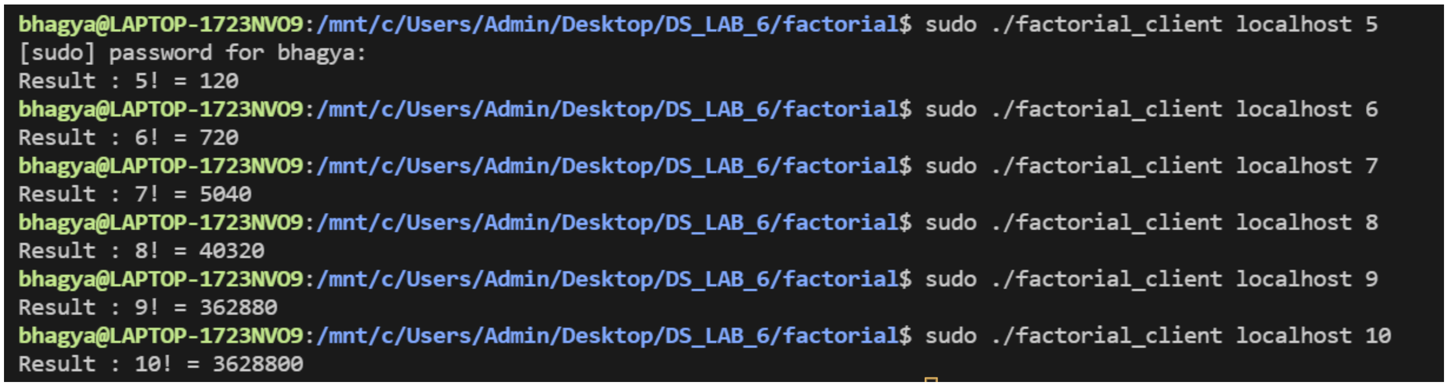
[Output]

Server:



```
bhagya@LAPTOP-1723NV09:/mnt/c/Users/Admin/Desktop/DS_LAB_6/factorial$ sudo ./factorial_server
Factorial of 5 is called
Factorial of 6 is called
Factorial of 7 is called
Factorial of 8 is called
Factorial of 9 is called
Factorial of 10 is called
```

Client:



```
bhagya@LAPTOP-1723NV09:/mnt/c/Users/Admin/Desktop/DS_LAB_6/factorial$ sudo ./factorial_client localhost 5
[sudo] password for bhagya:
Result : 5! = 120
bhagya@LAPTOP-1723NV09:/mnt/c/Users/Admin/Desktop/DS_LAB_6/factorial$ sudo ./factorial_client localhost 6
Result : 6! = 720
bhagya@LAPTOP-1723NV09:/mnt/c/Users/Admin/Desktop/DS_LAB_6/factorial$ sudo ./factorial_client localhost 7
Result : 7! = 5040
bhagya@LAPTOP-1723NV09:/mnt/c/Users/Admin/Desktop/DS_LAB_6/factorial$ sudo ./factorial_client localhost 8
Result : 8! = 40320
bhagya@LAPTOP-1723NV09:/mnt/c/Users/Admin/Desktop/DS_LAB_6/factorial$ sudo ./factorial_client localhost 9
Result : 9! = 362880
bhagya@LAPTOP-1723NV09:/mnt/c/Users/Admin/Desktop/DS_LAB_6/factorial$ sudo ./factorial_client localhost 10
Result : 10! = 3628800
```

2.) Implement **Calculator (Basic operation)**.

[calc.x]

```
struct numbers{
    int a;
    int b;
    char op;
};
program calc_PROG{
    version calc_VERS{
        int calc(numbers)=1;
    }=1;
}=0x4562877;
```

Run Command : rpcgen -a -C calc.x

- ✓ All required files will be created.
- ✓ The calc_client.c and calc_server.c files would be modified as following:

[calc_client.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 "calc.h"
void calc_prog_1(char *host)
{
    CLIENT *clnt;
    int *result_1;
    numbers calc_1_arg;
    char temp;

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

    printf("Enter 2 numbers: ");
    scanf("%d%d", &(calc_1_arg.a), &(calc_1_arg.b));
    scanf("%c", &temp);
```

```c
    printf("Press\n a for addition\n s for subtraction\n m for multiplication\n d for
division\n r for modulus\n Choice : ");
    scanf("%c", &(calc_1_arg.op));

    result_1 = calc_1(&calc_1_arg, clnt);
    if (result_1 == (int *)NULL)
    {
        clnt_perror(clnt, "call failed");
    }
    else
    {
        if (*result_1 == INT_MIN && calc_1_arg.op == 'd')
        {
            printf("Division by zero is not allowed\n");
        }
        else
        {
            printf("Result = %d\n", *result_1);
        }
    }
#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];
    calc_prog_1(host);
    exit(0);
}
```

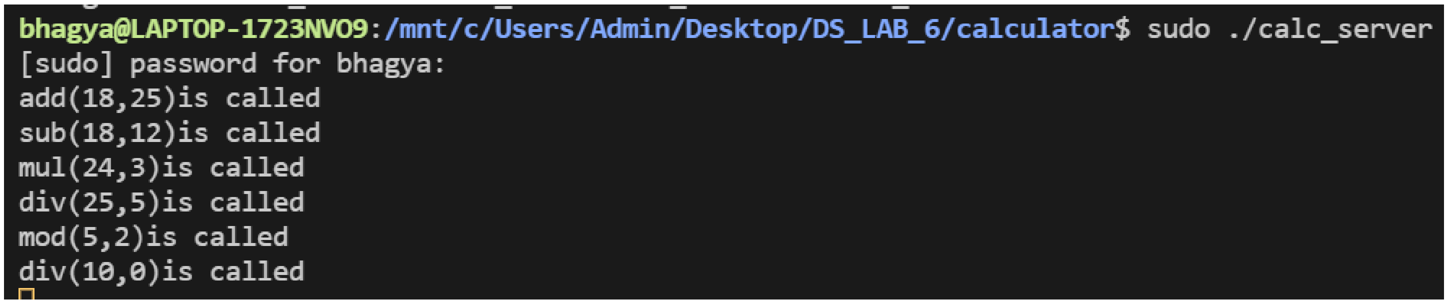
[calc_server.c]

```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 "calc.h"
int *calc_1_svc(numbers *argp, struct svc_req *rqstp)
{
    static int result;
    if (argp->op == 'a')
    {
        printf("add(%d,%d)is called\n", argp->a, argp->b);
        result = (argp->a) + (argp->b);
    }
    else if (argp->op == 's')
    {
        printf("sub(%d,%d)is called\n", argp->a, argp->b);
        result = (argp->a) - (argp->b);
    }
    else if (argp->op == 'm')
    {
        printf("mul(%d,%d)is called\n", argp->a, argp->b);
        result = (argp->a) * (argp->b);
    }
    else if (argp->op == 'r')
    {
        printf("mod(%d,%d)is called\n", argp->a, argp->b);
        result = (argp->a) % (argp->b);
    }
    else
    {
        printf("div(%d,%d)is called\n", argp->a, argp->b);
        if (argp->b != 0)
        {
            result = (argp->a) / (argp->b);
        }
        else
        {
            result = INT_MIN;
        }
    }
    return &result;
}
```

Run Command : make -f Makefile.calc

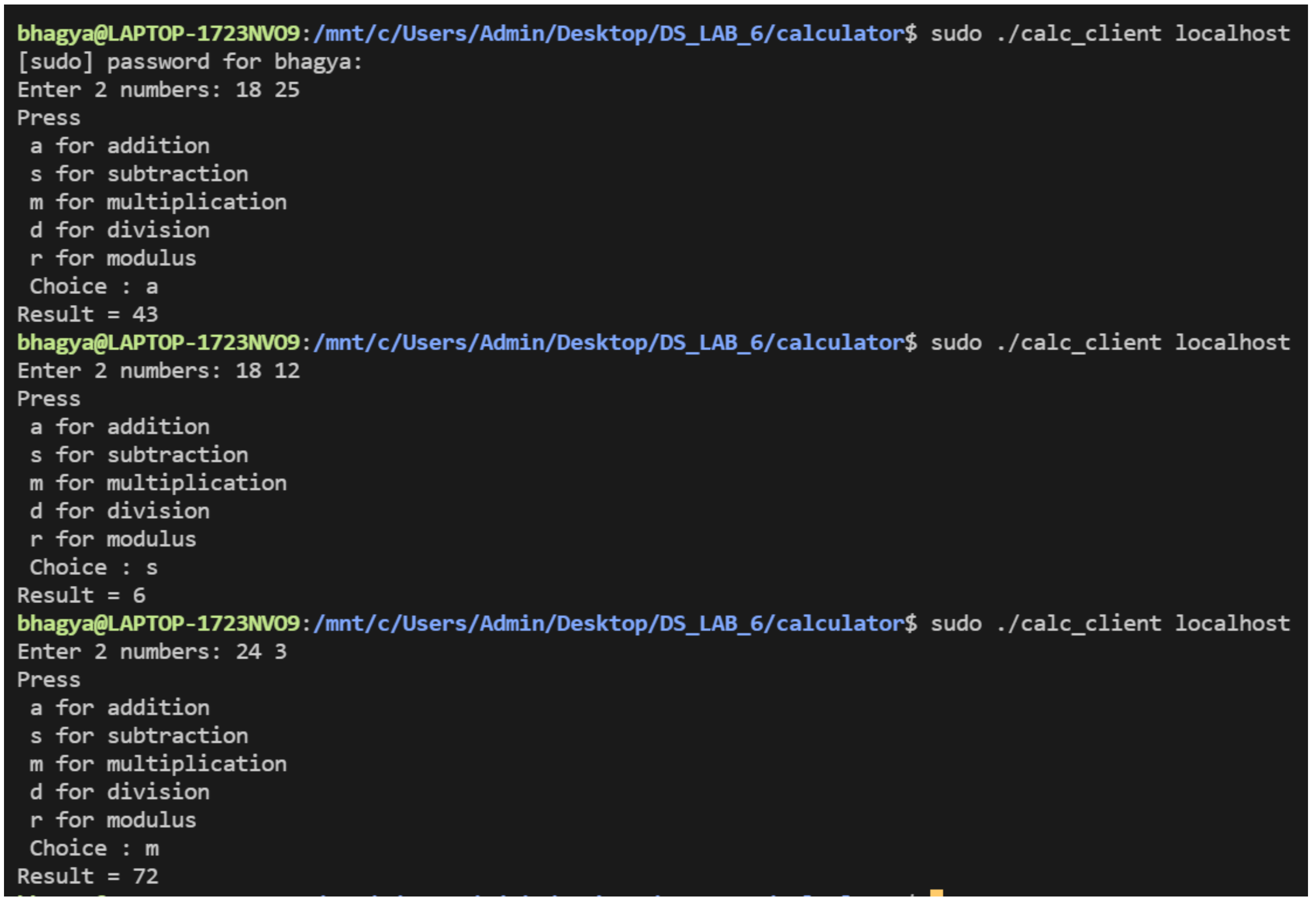
[Output]

Server:

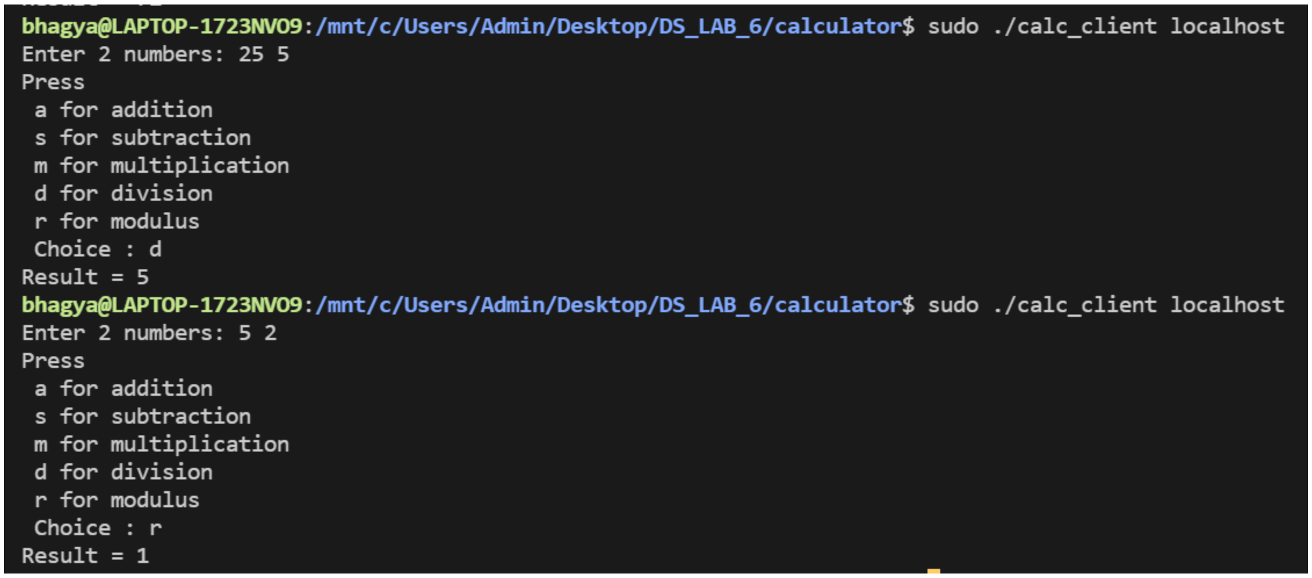


```
bhagya@LAPTOP-1723NVO9:/mnt/c/Users/Admin/Desktop/DS_LAB_6/calculator$ sudo ./calc_server
[sudo] password for bhagya:
add(18,25)is called
sub(18,12)is called
mul(24,3)is called
div(25,5)is called
mod(5,2)is called
div(10,0)is called
```

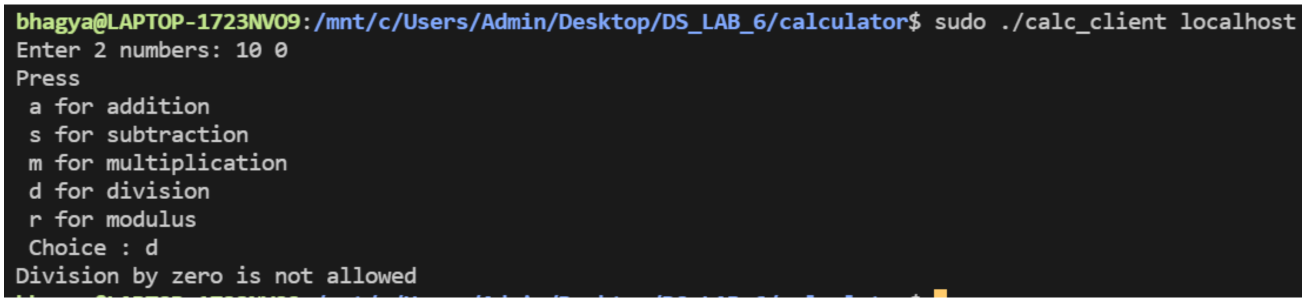
Client:



```
bhagya@LAPTOP-1723NV09:/mnt/c/Users/Admin/Desktop/DS_LAB_6/calculator$ sudo ./calc_client localhost
[sudo] password for bhagya:
Enter 2 numbers: 18 25
Press
 a for addition
 s for subtraction
 m for multiplication
 d for division
 r for modulus
 Choice : a
Result = 43
bhagya@LAPTOP-1723NV09:/mnt/c/Users/Admin/Desktop/DS_LAB_6/calculator$ sudo ./calc_client localhost
Enter 2 numbers: 18 12
Press
 a for addition
 s for subtraction
 m for multiplication
 d for division
 r for modulus
 Choice : s
Result = 6
bhagya@LAPTOP-1723NV09:/mnt/c/Users/Admin/Desktop/DS_LAB_6/calculator$ sudo ./calc_client localhost
Enter 2 numbers: 24 3
Press
 a for addition
 s for subtraction
 m for multiplication
 d for division
 r for modulus
 Choice : m
Result = 72
```



```
bhagya@LAPTOP-1723NV09:/mnt/c/Users/Admin/Desktop/DS_LAB_6/calculator$ sudo ./calc_client localhost
Enter 2 numbers: 25 5
Press
 a for addition
 s for subtraction
 m for multiplication
 d for division
 r for modulus
 Choice : d
Result = 5
bhagya@LAPTOP-1723NV09:/mnt/c/Users/Admin/Desktop/DS_LAB_6/calculator$ sudo ./calc_client localhost
Enter 2 numbers: 5 2
Press
 a for addition
 s for subtraction
 m for multiplication
 d for division
 r for modulus
 Choice : r
Result = 1
```



```
bhagya@LAPTOP-1723NV09:/mnt/c/Users/Admin/Desktop/DS_LAB_6/calculator$ sudo ./calc_client localhost
Enter 2 numbers: 10 0
Press
 a for addition
 s for subtraction
 m for multiplication
 d for division
 r for modulus
 Choice : d
Division by zero is not allowed
```

3.) Find out whether given number is **Prime** Number or not.

[prime.x]

```
program prime_PROG{
    version prime_VERS{
        int prime(int)=1;
    }=1;
}=0x4562877;
```

Run Command : rpcgen -a -C prime.x

- ✓ All required files will be created.
- ✓ The prime_client.c and prime_server.c files would be modified as following:

[prime_client.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 "prime.h"
void prime_prog_1(char *host)
{
    CLIENT *clnt;
    int *result_1;
    int prime_1_arg;

#ifndef DEBUG
    clnt = clnt_create(host, prime_PROG, prime_VERS, "udp");
    if (clnt == NULL)
    {
        clnt_pcreateerror(host);
        exit(1);
    }
#endif /* DEBUG */
    do
    {
        printf("Enter a non-zero postive number: ");
        scanf("%d", &prime_1_arg);
    } while (prime_1_arg < 1);
    result_1 = prime_1(&prime_1_arg, clnt);
    if (result_1 == (int *)NULL)
    {
        clnt_perror(clnt, "call failed");
```

```c
    }
    else
    {
        if (*result_1 == 0)
            printf("%d is not prime\n", prime_1_arg);
        else if (*result_1 == 1)
            printf("%d is prime\n", prime_1_arg);
        else
            printf("%d is neither prime nor composite\n", prime_1_arg);
    }
#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];
    prime_prog_1(host);
    exit(0);
}
```

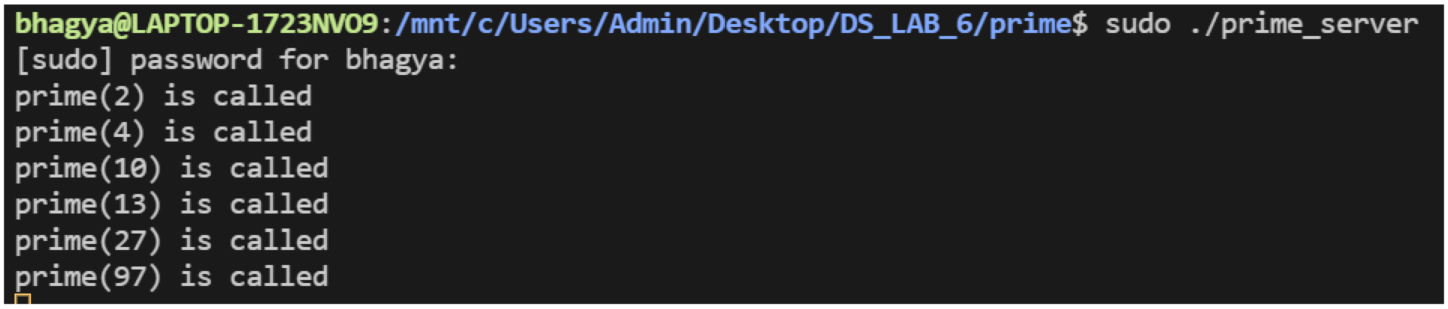
[prime_server.c]

```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 "prime.h"

int *prime_1_svc(int *argp, struct svc_req *rqstp)
{
    static int result;

    int n = *argp;
    printf("prime(%d) is called\n", n);

    if (n == 1)
    {
        result = 2;
        return &result;
    }
    result = 1;

    for (int i = 2; i <= n / 2; i++)
    {
        if (n % i == 0)
        {
            result = 0;
            break;
        }
    }
    return &result;
}
```

Run Command : make -f Makefile.prime

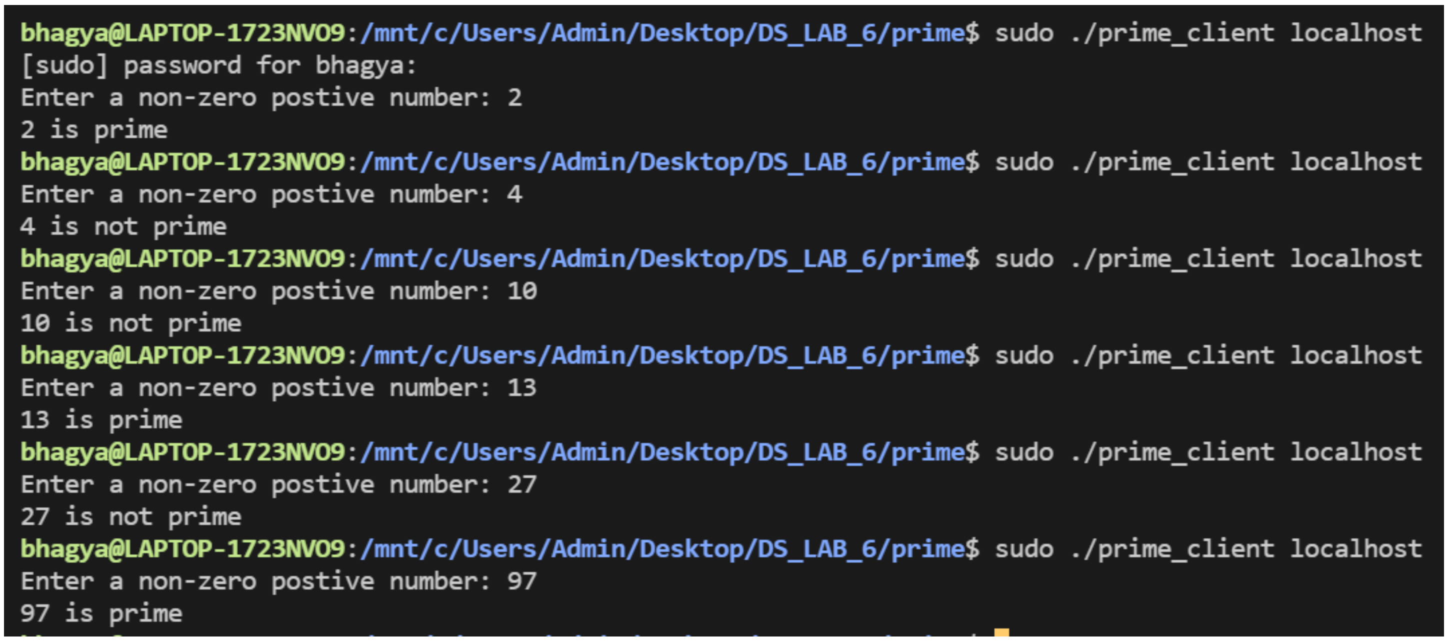
[Output]

Server:



```
bhagya@LAPTOP-1723NV09:/mnt/c/Users/Admin/Desktop/DS_LAB_6/prime$ sudo ./prime_server
[sudo] password for bhagya:
prime(2) is called
prime(4) is called
prime(10) is called
prime(13) is called
prime(27) is called
prime(97) is called
```

Client:



```
bhagya@LAPTOP-1723NV09:/mnt/c/Users/Admin/Desktop/DS_LAB_6/prime$ sudo ./prime_client localhost
[sudo] password for bhagya:
Enter a non-zero postive number: 2
2 is prime
bhagya@LAPTOP-1723NV09:/mnt/c/Users/Admin/Desktop/DS_LAB_6/prime$ sudo ./prime_client localhost
Enter a non-zero postive number: 4
4 is not prime
bhagya@LAPTOP-1723NV09:/mnt/c/Users/Admin/Desktop/DS_LAB_6/prime$ sudo ./prime_client localhost
Enter a non-zero postive number: 10
10 is not prime
bhagya@LAPTOP-1723NV09:/mnt/c/Users/Admin/Desktop/DS_LAB_6/prime$ sudo ./prime_client localhost
Enter a non-zero postive number: 13
13 is prime
bhagya@LAPTOP-1723NV09:/mnt/c/Users/Admin/Desktop/DS_LAB_6/prime$ sudo ./prime_client localhost
Enter a non-zero postive number: 27
27 is not prime
bhagya@LAPTOP-1723NV09:/mnt/c/Users/Admin/Desktop/DS_LAB_6/prime$ sudo ./prime_client localhost
Enter a non-zero postive number: 97
97 is prime
```

4.) Print out the **Fibonacci series** till the given number.

[fib.x]

```
struct sequence{
    int a[50];
};
program FIB_PROG{
    version FIB_VERS{
        sequence fib(int)=1;
    }=1;
}=0x4562877;
```

Run Command : rpcgen –a –C fib.x

- ✓ All required files will be created.
- ✓ The prime_client.c and prime_server.c files would be modified as following:

[fib_client.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 "fib.h"

void fib_prog_1(char *host, int x)
{
    CLIENT *clnt;
    sequence *result_1;
    int fib_1_arg;

#ifndef DEBUG
    clnt = clnt_create(host, FIB_PROG, FIB_VERS, "udp");
    if (clnt == NULL)
    {
        clnt_pcreateerror(host);
        exit(1);
    }
#endif /* DEBUG */
    fib_1_arg = x;
    result_1 = fib_1(&fib_1_arg, clnt);
    if (result_1 == (sequence *)NULL)
    {
```

```c
        clnt_perror(clnt, "call failed");
    }
    else
    {
        printf("Fib series upto %d: ", fib_1_arg);
        int i = 0;
        while (1)
        {
            if (result_1->a[i] < fib_1_arg)
                printf("%d ", result_1->a[i++]);
            else
                break;
        }
        printf("\n");
    }
#ifndef DEBUG
    clnt_destroy(clnt);
#endif /* DEBUG */
}

int main(int argc, char *argv[])
{
    char *host;

    if (argc < 3)
    {
        printf("usage: %s server_host\n", argv[0]);
        exit(1);
    }
    host = argv[1];
    fib_prog_1(host, atoi(argv[2]));
    exit(0);
}
```

[fib_server.c]

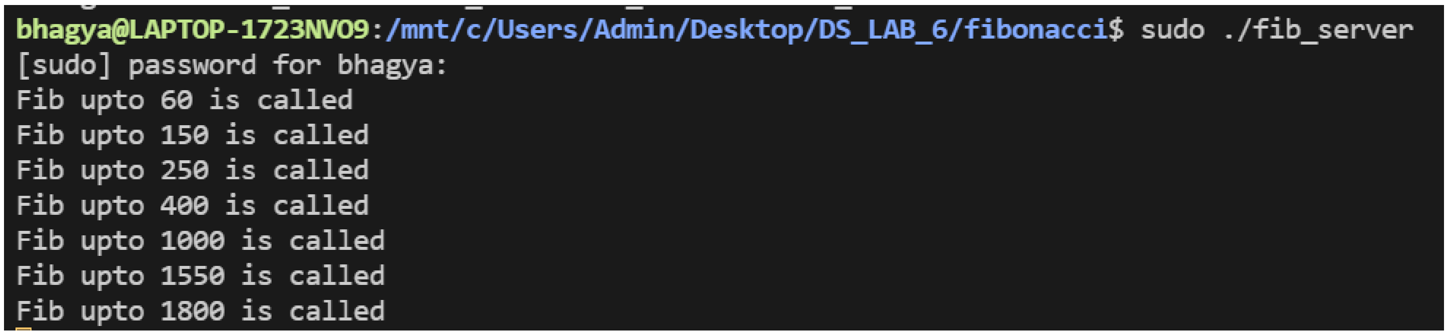
```c
/*
 * This is sample code generated by rpcgen.
 * These are only templates and you can use them
 * as a guideline for developing your own functions.
 */

#include "fib.h"

sequence *
fib_1_svc(int *argp, struct svc_req *rqstp)
```

```
{
    static sequence result;

    result.a[0] = 0;
    result.a[1] = 1;
    int i = 2;
    printf("Fib upto %d is called\n", *argp);
    while (result.a[i - 1] < (*argp))
    {
        result.a[i] = result.a[i - 1] + result.a[i - 2];
        i++;
    }
    return &result;
}
```

Run Command : make -f Makefile.fib

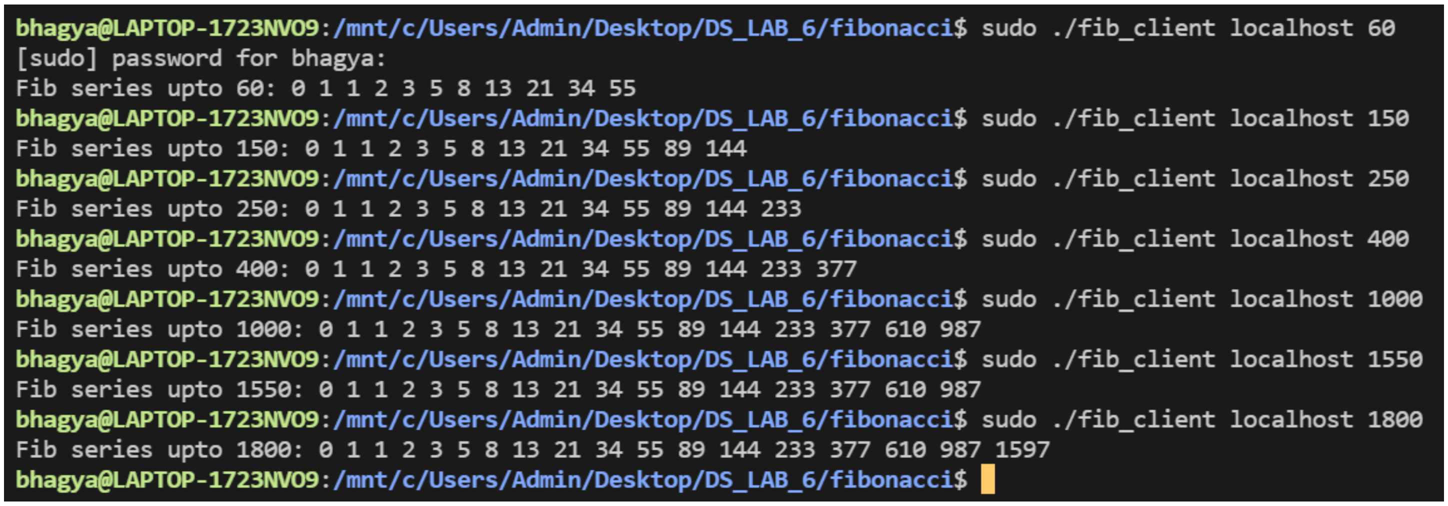
[Output]

Server:



```
bhagya@LAPTOP-1723NV09:/mnt/c/Users/Admin/Desktop/DS_LAB_6/fibonacci$ sudo ./fib_server
[sudo] password for bhagya:
Fib upto 60 is called
Fib upto 150 is called
Fib upto 250 is called
Fib upto 400 is called
Fib upto 1000 is called
Fib upto 1550 is called
Fib upto 1800 is called
```

Client:



```
bhagya@LAPTOP-1723NV09:/mnt/c/Users/Admin/Desktop/DS_LAB_6/fibonacci$ sudo ./fib_client localhost 60
[sudo] password for bhagya:
Fib series upto 60: 0 1 1 2 3 5 8 13 21 34 55
bhagya@LAPTOP-1723NV09:/mnt/c/Users/Admin/Desktop/DS_LAB_6/fibonacci$ sudo ./fib_client localhost 150
Fib series upto 150: 0 1 1 2 3 5 8 13 21 34 55 89 144
bhagya@LAPTOP-1723NV09:/mnt/c/Users/Admin/Desktop/DS_LAB_6/fibonacci$ sudo ./fib_client localhost 250
Fib series upto 250: 0 1 1 2 3 5 8 13 21 34 55 89 144 233
bhagya@LAPTOP-1723NV09:/mnt/c/Users/Admin/Desktop/DS_LAB_6/fibonacci$ sudo ./fib_client localhost 400
Fib series upto 400: 0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
bhagya@LAPTOP-1723NV09:/mnt/c/Users/Admin/Desktop/DS_LAB_6/fibonacci$ sudo ./fib_client localhost 1000
Fib series upto 1000: 0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
bhagya@LAPTOP-1723NV09:/mnt/c/Users/Admin/Desktop/DS_LAB_6/fibonacci$ sudo ./fib_client localhost 1550
Fib series upto 1550: 0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
bhagya@LAPTOP-1723NV09:/mnt/c/Users/Admin/Desktop/DS_LAB_6/fibonacci$ sudo ./fib_client localhost 1800
Fib series upto 1800: 0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597
bhagya@LAPTOP-1723NV09:/mnt/c/Users/Admin/Desktop/DS_LAB_6/fibonacci$
```

5.) Find the **Maximum** value of an array of integers using RPC.

[maxArray.x]

```
struct numbers{
    int array[100];
    int size;
};
program maxArray_PROG{
    version maxArray_VERS{
        int maxArray(numbers)=1;
    }=1;
}=0x4562877;
```

Run Command : rpcgen -a -C **maxArray**.x

- ✓ All required files will be created.
- ✓ The maxArray_client.c and maxArray _server.c files would be modified as following:

[maxArray_client.c]

```
#include "maxArray.h"

void maxarray_prog_1(char *host)
{
    CLIENT *clnt;
    int *result_1;
    numbers maxarray_1_arg;

#ifndef DEBUG
    clnt = clnt_create(host, maxArray_PROG, maxArray_VERS, "udp");
    if (clnt == NULL)
    {
        clnt_pcreateerror(host);
        exit(1);
    }
#endif /* DEBUG */
    printf("Enter the number of elements in the array : ");
    scanf("%d", &maxarray_1_arg.size);
    printf("Enter the elements of the array : \n");
    for (int i = 0; i < maxarray_1_arg.size; i++)
    {
        scanf("%d", &maxarray_1_arg.array[i]);
    }
    result_1 = maxarray_1(&maxarray_1_arg, clnt);
    if (result_1 == (int *)NULL)
    {
```

```c
        clnt_perror(clnt, "call failed");
    }
    else
    {
        printf("The maximum element in the array is : %d\n", *result_1);
    }
#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];
    maxarray_prog_1(host);
    exit(0);
}
```

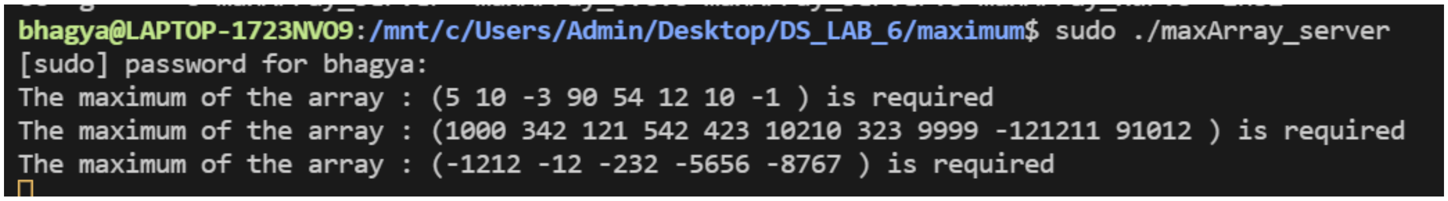
[maxArray_server.c]

```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 "maxArray.h"

int *maxarray_1_svc(numbers *argp, struct svc_req *rqstp)
{
    static int result;

    printf("The maximum of the array : (");
    for (int i = 0; i < argp->size; i++)
    {
        printf("%d ", argp->array[i]);
    }
    printf(") is required\n");
    int n = argp->size;
    int *arr = argp->array;
    int max = arr[0];
    for (int i = 1; i < n; i++)
    {
        if (arr[i] > max)
        {
            max = arr[i];
        }
    }
    result = max;

    return &result;
}
```

Run Command : make -f Makefile.maxArray

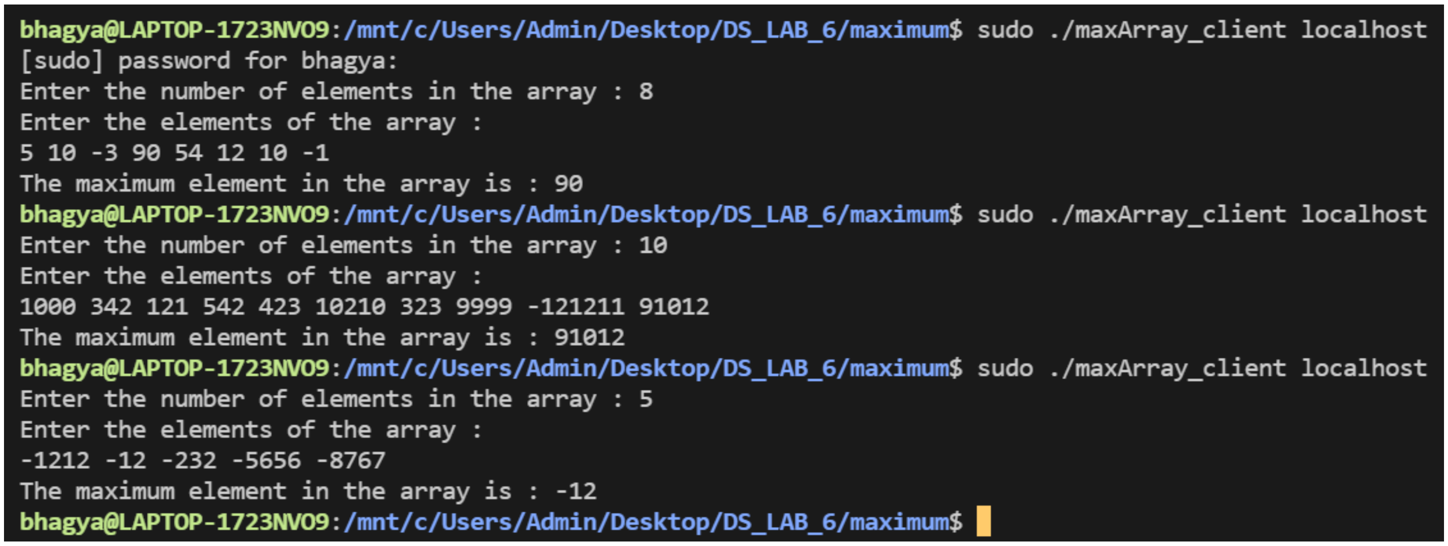
[Output]

Server:



```
bhagya@LAPTOP-1723NV09:/mnt/c/Users/Admin/Desktop/DS_LAB_6/maximum$ sudo ./maxArray_server
[sudo] password for bhagya:
The maximum of the array : (5 10 -3 90 54 12 10 -1 ) is required
The maximum of the array : (1000 342 121 542 423 10210 323 9999 -121211 91012 ) is required
The maximum of the array : (-1212 -12 -232 -5656 -8767 ) is required
```

Client:



```
bhagya@LAPTOP-1723NV09:/mnt/c/Users/Admin/Desktop/DS_LAB_6/maximum$ sudo ./maxArray_client localhost
[sudo] password for bhagya:
Enter the number of elements in the array : 8
Enter the elements of the array :
5 10 -3 90 54 12 10 -1
The maximum element in the array is : 90
bhagya@LAPTOP-1723NV09:/mnt/c/Users/Admin/Desktop/DS_LAB_6/maximum$ sudo ./maxArray_client localhost
Enter the number of elements in the array : 10
Enter the elements of the array :
1000 342 121 542 423 10210 323 9999 -121211 91012
The maximum element in the array is : 91012
bhagya@LAPTOP-1723NV09:/mnt/c/Users/Admin/Desktop/DS_LAB_6/maximum$ sudo ./maxArray_client localhost
Enter the number of elements in the array : 5
Enter the elements of the array :
-1212 -12 -232 -5656 -8767
The maximum element in the array is : -12
bhagya@LAPTOP-1723NV09:/mnt/c/Users/Admin/Desktop/DS_LAB_6/maximum$
```

SUBMITTED BY: U19CS012

BHAGYA VINOD RANA