DS Assignment 5

Name: Himani Verma

Admission No: U19CS075

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.

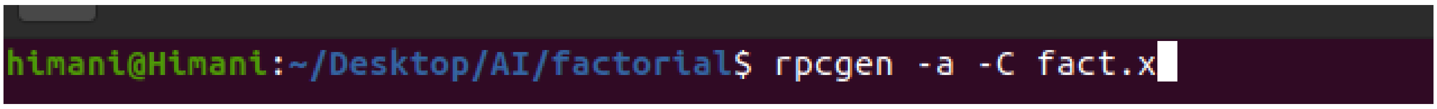
Steps:

First create a fact.x file. Navigate to its location and write the following commands.

Source Code for fact.x:

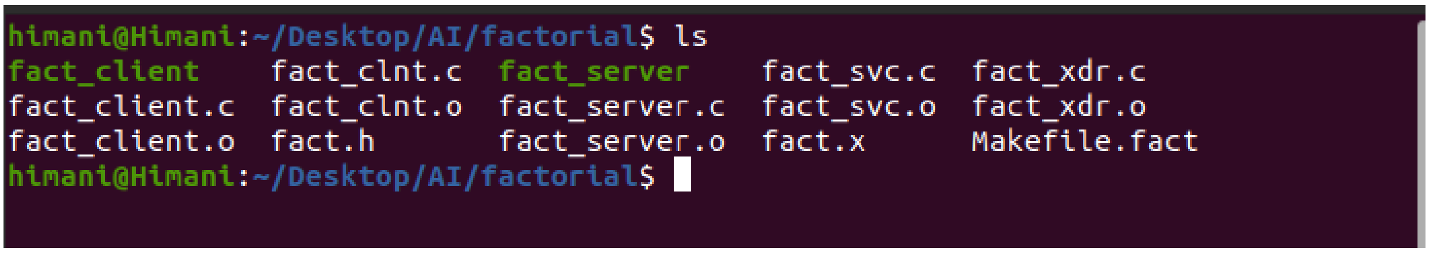
```
struct intpair {
    int a;
};

program FACT_PROG
{
 version FACT_VERS
 {
  int FACT(intpair) = 1;
 } = 1;

} = 0x23451111;
```

Now execute the below in terminal:



```
himani@Himani:~/Desktop/AI/factorial$ rpcgen -a -C fact.x
```

This will create the required files:



```
himani@Himani:~/Desktop/AI/factorial$ ls
fact_client     fact_clnt.c  fact_server     fact_svc.c  fact_xdr.c
fact_client.c   fact_clnt.o  fact_server.c   fact_svc.o  fact_xdr.o
fact_client.o   fact.h       fact_server.o   fact.x      Makefile.fact
himani@Himani:~/Desktop/AI/factorial$
```

Now replace the following commands in Makefile.fact

CFLAGS += -g to: **CFLAGS += -g -DRPC_SVC_FG**

And

RPCGENFLAGS = to: **RPCGENFLAGS = -C**

Source Code fact_client.c file:

```
#include "fact.h"
void
```

```c
fact_prog_1(char *host,int a)
{
    CLIENT *clnt;
    int  *result_1;
    intpair  fact_1_arg;

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

    fact_1_arg.a = a;
    result_1 = fact_1(&fact_1_arg, clnt);
    if (result_1 == (int *) NULL) {
        clnt_perror (clnt, "call failed");
    }else{
        printf("FACTORIAL : %d\n", *result_1);
    }
#ifndef DEBUG
    clnt_destroy (clnt);
#endif   /* DEBUG */
}

int main (int argc, char *argv[])
{
    char *host;
    int num;
    if (argc < 2) {
        printf ("usage: %s server_host\n", argv[0]);
        exit (1);
    }
    host = argv[1];
    printf("Enter the Number: ");
    scanf("%d",&num);
    fact_prog_1 (host,num);
exit (0);
}
```

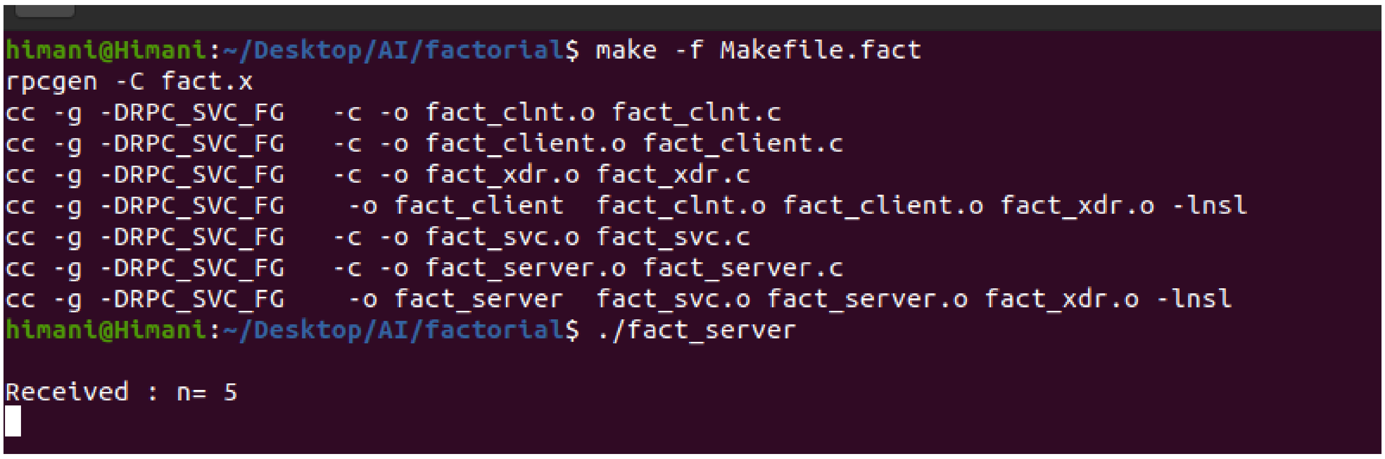
Source Code fact_server.c file:

```c
#include "fact.h"

int * fact_1_svc(intpair *argp, struct svc_req *rqstp)
{
    static int  result,n,fact;

    /*
     * insert server code here
     */
     int i;
     n=argp->a;
     // factorial logic
     fact = 1;
     printf("\nReceived : n= %d \n",n);

     for (i=n;i>0;i--)
     {
      fact=fact * i;
     }
     result=fact;
    return &result;
}
```

Now execute the below in terminal:

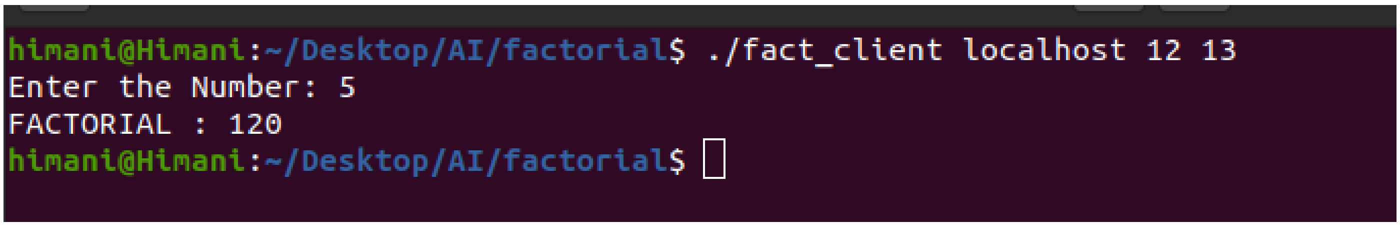
Output:

Server Side:



```
himani@Himani:~/Desktop/AI/factorial$ make -f Makefile.fact
rpcgen -C fact.x
cc -g -DRPC_SVC_FG   -c -o fact_clnt.o fact_clnt.c
cc -g -DRPC_SVC_FG   -c -o fact_client.o fact_client.c
cc -g -DRPC_SVC_FG   -c -o fact_xdr.o fact_xdr.c
cc -g -DRPC_SVC_FG    -o fact_client  fact_clnt.o fact_client.o fact_xdr.o -lnsl
cc -g -DRPC_SVC_FG   -c -o fact_svc.o fact_svc.c
cc -g -DRPC_SVC_FG   -c -o fact_server.o fact_server.c
cc -g -DRPC_SVC_FG    -o fact_server  fact_svc.o fact_server.o fact_xdr.o -lnsl
himani@Himani:~/Desktop/AI/factorial$ ./fact_server

Received : n= 5
```

Client Side:



```
himani@Himani:~/Desktop/AI/factorial$ ./fact_client localhost 12 13
Enter the Number: 5
FACTORIAL : 120
himani@Himani:~/Desktop/AI/factorial$
```

2. Implement Calculator (Basic operation).

Source Code for calc.x file

```
struct intpair {
    int a;
    int b;
    int ch;
};

program CALC_PROG
{
 version CALC_VERS
 {
  int CALC(intpair) = 1;
 } = 1;

} = 0x23451111;
```

Source Code for calc_client.c file

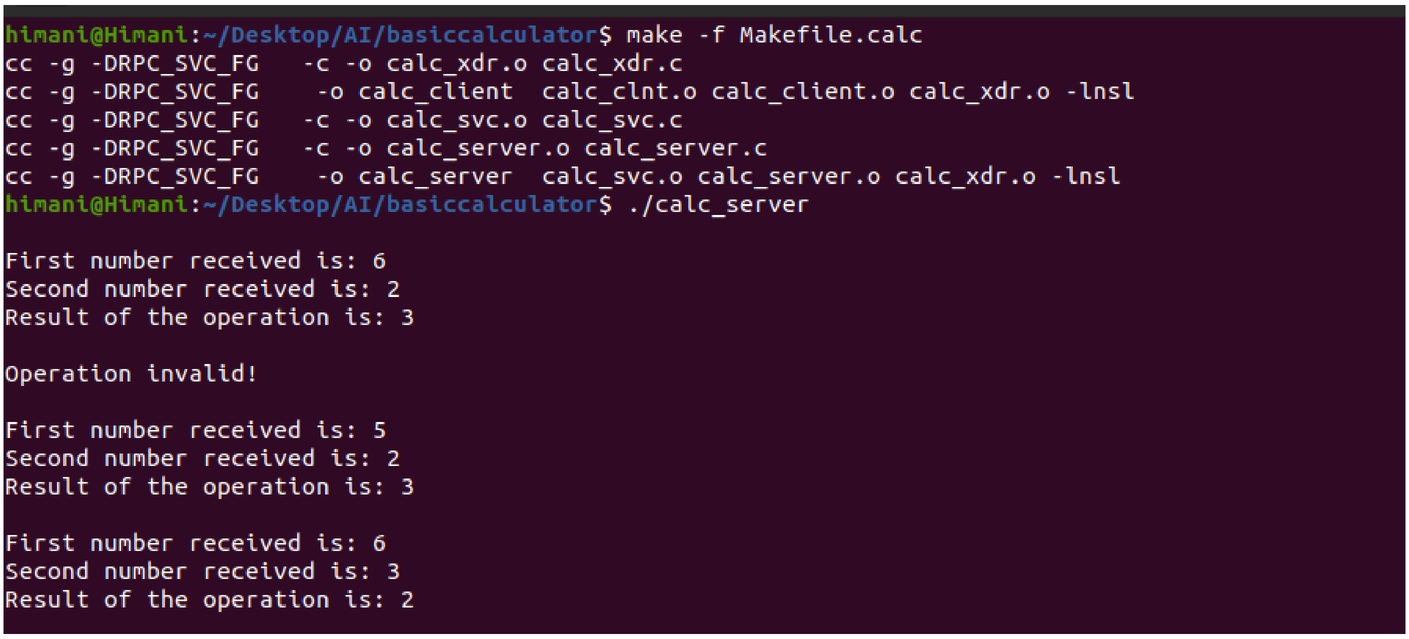
```
#include "calc.h"
void calc_prog_1(char *host,int x,int y, int choice){
    CLIENT *clnt;
    int  *result_1;
    intpair  calc_1_arg;


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

    calc_1_arg.a=x;
    calc_1_arg.b=y;
    calc_1_arg.ch=choice;
    result_1 = calc_1(&calc_1_arg, clnt);
    if (result_1 == (int *) NULL) {
        clnt_perror (clnt, "call failed");
    }
    else{
        printf("Result : %d\n", *result_1);
    }
#ifndef DEBUG
    clnt_destroy (clnt);
#endif   /* DEBUG */
}


int main (int argc, char *argv[])
{
    char *host;
    int num1,num2,choice;
    if (argc < 2) {
        printf ("usage: %s server_host\n", argv[0]);
        exit (1);
    }
    host = argv[1];
    printf("Enter First Number: ");
    scanf("%d", &num1);
    printf("Enter Second Number: ");
    scanf("%d", &num2);
    printf("Enter the Operation you want to perform: ");
    printf("\n1. Addition\n2.Subtraction\n3.Multiplication\n4.Division\n");
    scanf("%d", &choice);
    calc_prog_1 (host,num1,num2,choice);
    exit (0);
}
```

Source Code for calc_server.c file

```
#include "calc.h"

int * calc_1_svc(intpair *argp, struct svc_req *rqstp)
{
    static int  result;
    int x,y,op;
    x=argp->a;
    y=argp->b;
    op=argp->ch;

    switch(op){
        case 1:
            result=x+y;
            break;
        case 2:
            result=x-y;
            break;
        case 3:
            result=x*y;
            break;
        case 4:
            if(y==0){
                printf("\nOperation invalid!\n");
                result=-1;
                return &result;
            }else{
                result=x/y;
            }
            break;
        default:
            printf("\nWrong Choice! Enter between 1 to 4\n");
            result=-1;
            return &result;
            break;
    }

    printf("\nFirst number received is: %d\n",x);
    printf("Second number received is: %d\n",y);
    printf("Result of the operation is: %d\n",result);
    return &result;
}
```

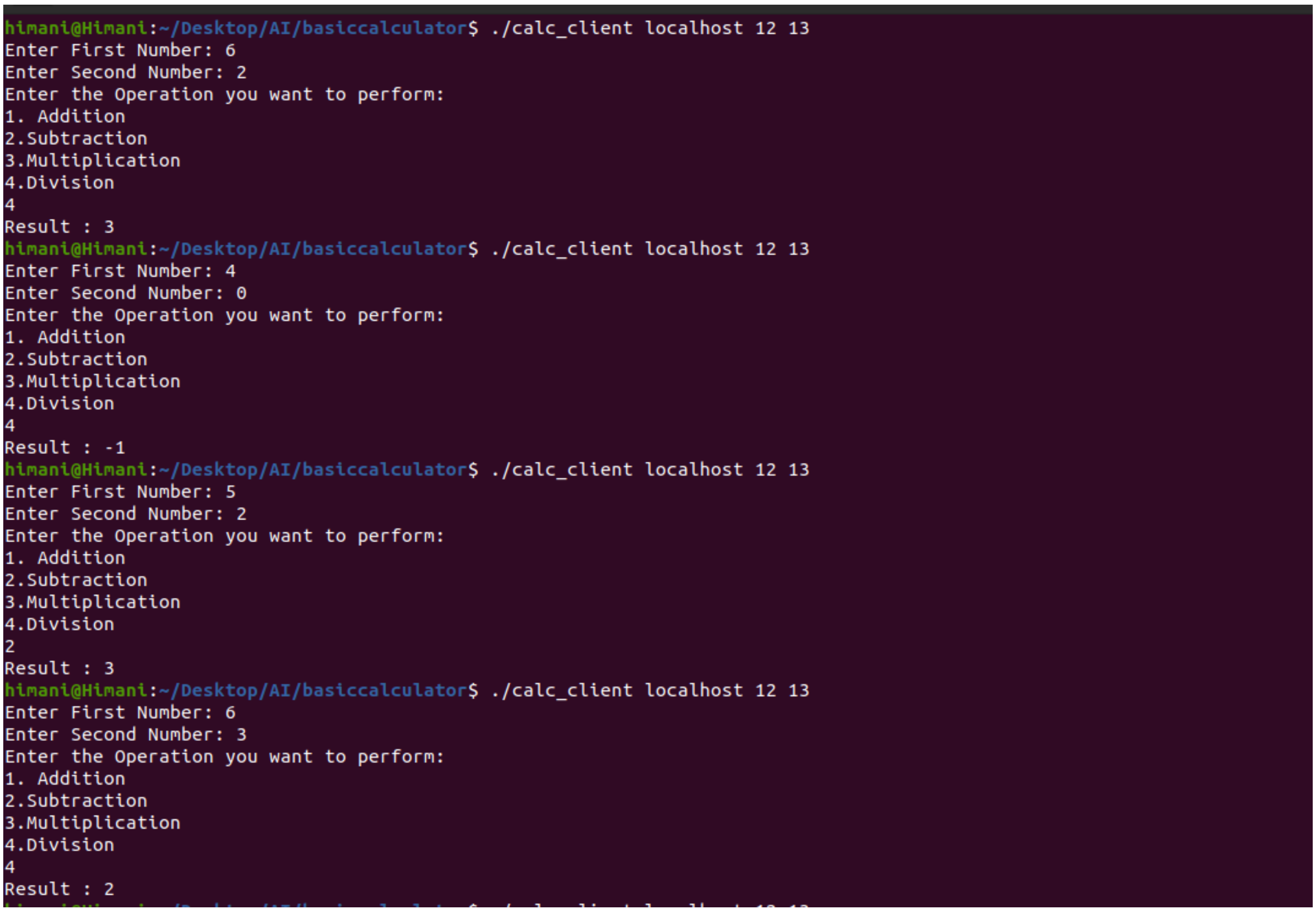
Output:

Server Side:



```
himani@Himani:~/Desktop/AI/basiccalculator$ make -f Makefile.calc
cc -g -DRPC_SVC_FG   -c -o calc_xdr.o calc_xdr.c
cc -g -DRPC_SVC_FG    -o calc_client  calc_clnt.o calc_client.o calc_xdr.o -lnsl
cc -g -DRPC_SVC_FG   -c -o calc_svc.o calc_svc.c
cc -g -DRPC_SVC_FG   -c -o calc_server.o calc_server.c
cc -g -DRPC_SVC_FG    -o calc_server  calc_svc.o calc_server.o calc_xdr.o -lnsl
himani@Himani:~/Desktop/AI/basiccalculator$ ./calc_server

First number received is: 6
Second number received is: 2
Result of the operation is: 3

Operation invalid!

First number received is: 5
Second number received is: 2
Result of the operation is: 3

First number received is: 6
Second number received is: 3
Result of the operation is: 2
```

Client Side:



```
himani@Himani:~/Desktop/AI/basiccalculator$ ./calc_client localhost 12 13
Enter First Number: 6
Enter Second Number: 2
Enter the Operation you want to perform:
1. Addition
2.Subtraction
3.Multiplication
4.Division
4
Result : 3
himani@Himani:~/Desktop/AI/basiccalculator$ ./calc_client localhost 12 13
Enter First Number: 4
Enter Second Number: 0
Enter the Operation you want to perform:
1. Addition
2.Subtraction
3.Multiplication
4.Division
4
Result : -1
himani@Himani:~/Desktop/AI/basiccalculator$ ./calc_client localhost 12 13
Enter First Number: 5
Enter Second Number: 2
Enter the Operation you want to perform:
1. Addition
2.Subtraction
3.Multiplication
4.Division
2
Result : 3
himani@Himani:~/Desktop/AI/basiccalculator$ ./calc_client localhost 12 13
Enter First Number: 6
Enter Second Number: 3
Enter the Operation you want to perform:
1. Addition
2.Subtraction
3.Multiplication
4.Division
4
Result : 2
```

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

Source Code for prime.x file

```
struct intpair {
    int a;
};
```

```
program PRIME_PROG
{
 version PRIME_VERS
 {
  int PRIME(intpair) = 1;
 } = 1;

} = 0x23451111;
```

Source Code for prime_client.c file

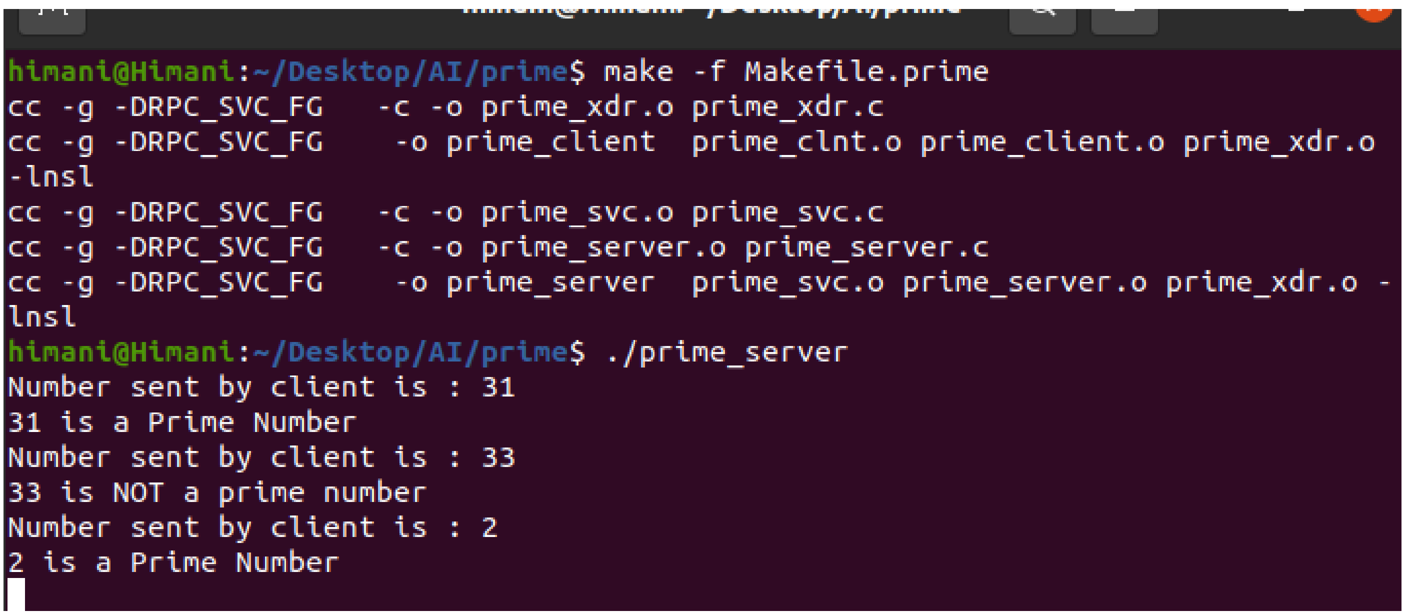
```c
#include "prime.h"

void prime_prog_1(char *host,int num)
{
    CLIENT *clnt;
    int  *result_1;
    intpair  prime_1_arg;

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

    prime_1_arg.a=num;
    result_1 = prime_1(&prime_1_arg, clnt);
    if (result_1 == (int *) NULL) {
        clnt_perror (clnt, "call failed");
    }
    else{
        if(*result_1==0){
            printf("It is a Prime Number\n");
        }else{
            printf("It is Not a Prime Number\n");
        }
    }
#ifndef DEBUG
    clnt_destroy (clnt);
#endif   /* DEBUG */
}

int main (int argc, char *argv[])
{
    char *host;
    int num;
    if (argc < 2) {
        printf ("usage: %s server_host\n", argv[0]);
        exit (1);
    }
    host = argv[1];
    printf("Enter the Number: ");
    scanf("%d",&num);
    prime_prog_1 (host,num);
    exit (0);
}
```

Source Code for prime_server.c file

```
#include "prime.h"

int * prime_1_svc(intpair *argp, struct svc_req *rqstp)
{
    static int  result,n;
    int i,flag=0;
    n=argp->a;
    printf("Number sent by client is : %d\n",n);
    for(int i=2; i<n/2; i++){
    if(n%i==0){
        printf("%d is NOT a prime number\n", n);
        flag=1;
        break;
     }
    }
    if(flag==0){
       printf("%d is a Prime Number\n", n);
     }
    result=flag;
    return &result;
}
```

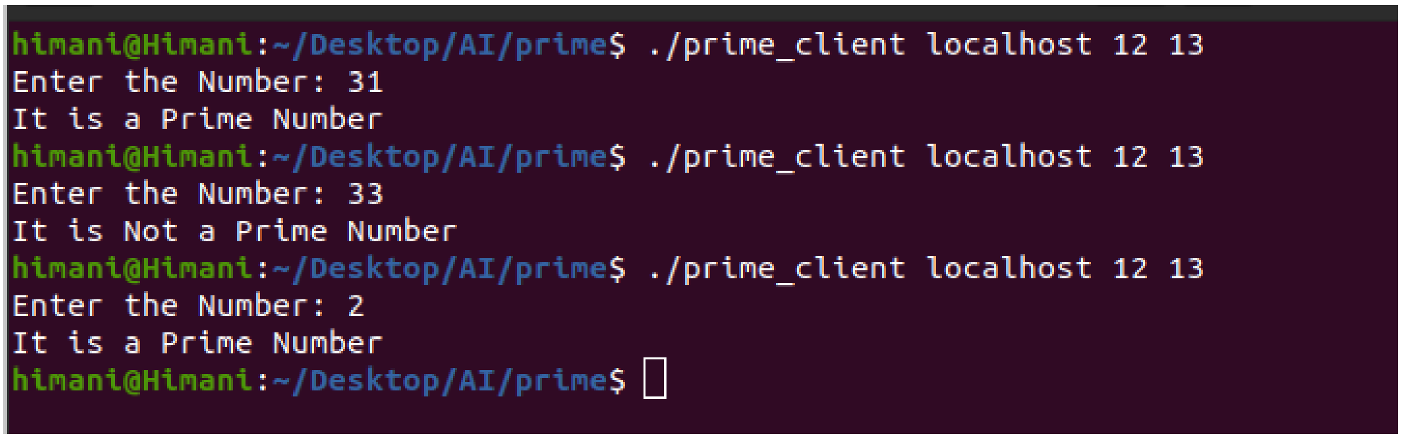
Output:

Server Side:



```
himani@Himani:~/Desktop/AI/prime$ make -f Makefile.prime
cc -g -DRPC_SVC_FG   -c -o prime_xdr.o prime_xdr.c
cc -g -DRPC_SVC_FG    -o prime_client  prime_clnt.o prime_client.o prime_xdr.o
-lnsl
cc -g -DRPC_SVC_FG   -c -o prime_svc.o prime_svc.c
cc -g -DRPC_SVC_FG   -c -o prime_server.o prime_server.c
cc -g -DRPC_SVC_FG    -o prime_server  prime_svc.o prime_server.o prime_xdr.o -
lnsl
himani@Himani:~/Desktop/AI/prime$ ./prime_server
Number sent by client is : 31
31 is a Prime Number
Number sent by client is : 33
33 is NOT a prime number
Number sent by client is : 2
2 is a Prime Number
```

Client Side:



```
himani@Himani:~/Desktop/AI/prime$ ./prime_client localhost 12 13
Enter the Number: 31
It is a Prime Number
himani@Himani:~/Desktop/AI/prime$ ./prime_client localhost 12 13
Enter the Number: 33
It is Not a Prime Number
himani@Himani:~/Desktop/AI/prime$ ./prime_client localhost 12 13
Enter the Number: 2
It is a Prime Number
himani@Himani:~/Desktop/AI/prime$ 
```

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

Source Code for fib.x file

```
struct intpair {
    int a;
};

struct newpair {
    int b[200];
};

program FIB_PROG
{
    version FIB_VERS
    {
        newpair FIB(intpair) = 1;
    } = 1;

} = 0x23451111;
```

Source Code for fib_client.c file

```
#include "fib.h"

void fib_prog_1(char *host,int num)
{
    CLIENT *clnt;
    newpair  *result_1;
    intpair  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.a = num;
    result_1 = fib_1(&fib_1_arg, clnt);
    if (result_1 == (newpair *) NULL) {
        clnt_perror (clnt, "call failed");
```

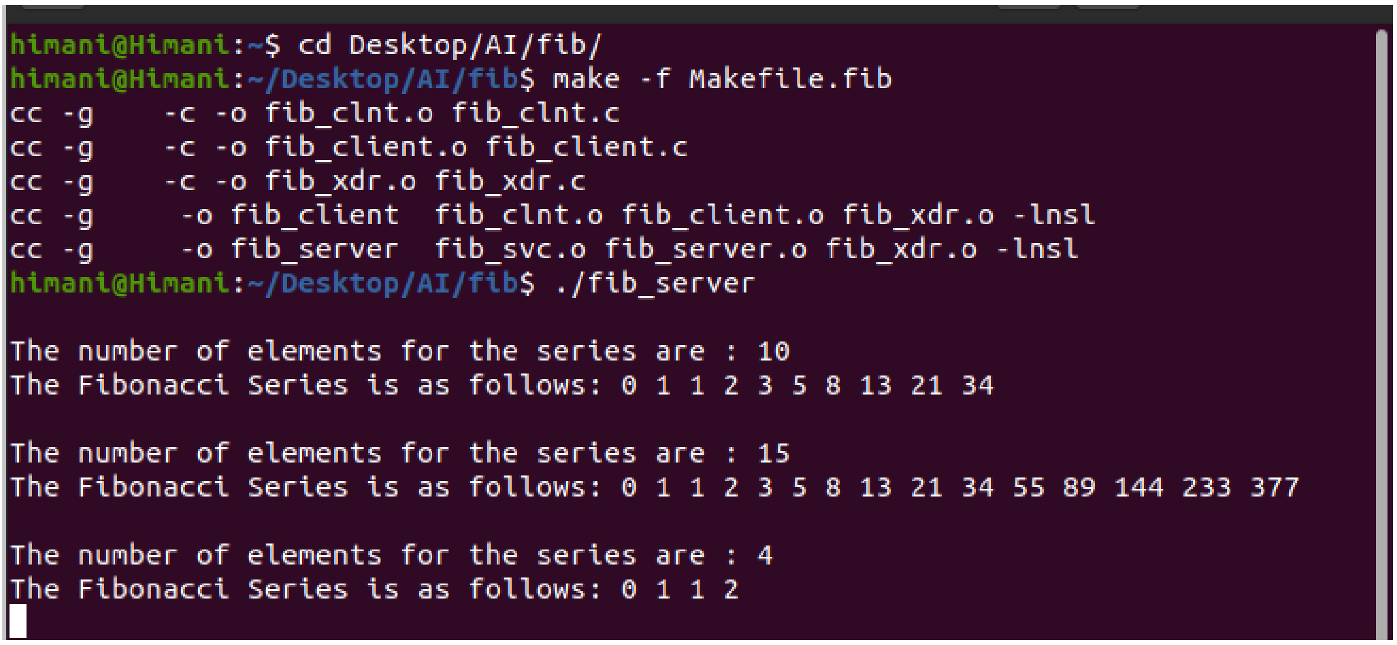
```c
    }else{
        int i = 0;
        printf("The Fibonacci Series is as follows: ");
        for(i=0;i<num;i++){
            printf("%d ",result_1->b[i]);
        }
        printf("\n");
    }

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

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

    if (argc < 2) {
        printf ("usage: %s server_host\n", argv[0]);
        exit (1);
    }
    host = argv[1];
    printf("Enter the number of series elements: ");
    scanf("%d",&num);
    if(num>200) exit(0);
    fib_prog_1 (host,num);
    exit (0);
}
```

Source Code for fib_server.c file

```c
#include "fib.h"

newpair * fib_1_svc(intpair *argp, struct svc_req *rqstp)
{
    static newpair result;
    int n = argp->a;

    int a0=0,a1=1,i;
    printf("\nThe number of elements for the series are : %d",n);
    printf("\nThe Fibonacci Series is as follows: ");

    for(i=0;i<n;i++){
        if(i==0){
            result.b[i]=a0;
        }else if(i==1){
            result.b[i]=a1;
        }else{
            result.b[i]= a0 + a1;
            a0 = a1;
            a1 = result.b[i];
        }
        printf("%d ",result.b[i]);
    }

    printf("\n");
    return &result;
}
```

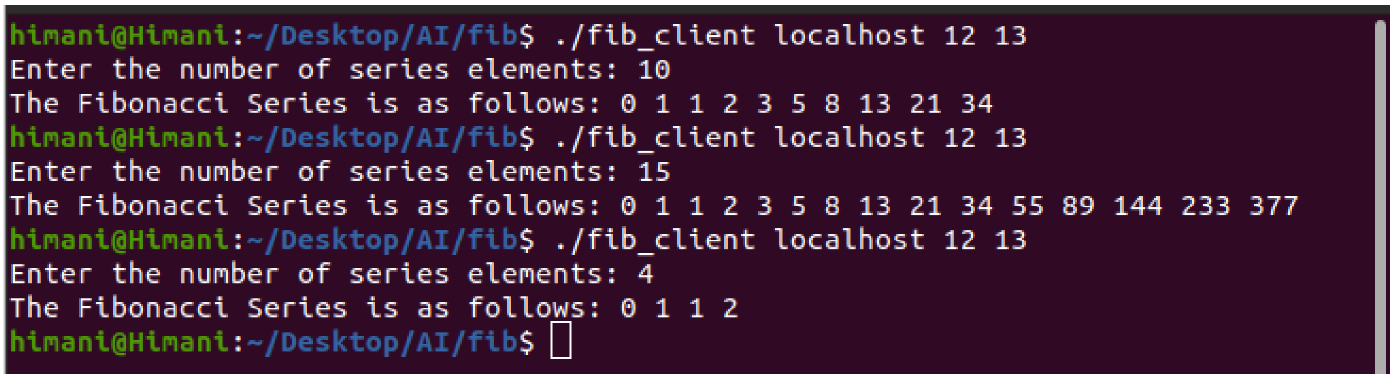
Output:

Server Side:



```
himani@Himani:~$ cd Desktop/AI/fib/
himani@Himani:~/Desktop/AI/fib$ make -f Makefile.fib
cc -g    -c -o fib_clnt.o fib_clnt.c
cc -g    -c -o fib_client.o fib_client.c
cc -g    -c -o fib_xdr.o fib_xdr.c
cc -g     -o fib_client  fib_clnt.o fib_client.o fib_xdr.o -lnsl
cc -g     -o fib_server  fib_svc.o fib_server.o fib_xdr.o -lnsl
himani@Himani:~/Desktop/AI/fib$ ./fib_server

The number of elements for the series are : 10
The Fibonacci Series is as follows: 0 1 1 2 3 5 8 13 21 34

The number of elements for the series are : 15
The Fibonacci Series is as follows: 0 1 1 2 3 5 8 13 21 34 55 89 144 233 377

The number of elements for the series are : 4
The Fibonacci Series is as follows: 0 1 1 2
```

Client Side:



```
himani@Himani:~/Desktop/AI/fib$ ./fib_client localhost 12 13
Enter the number of series elements: 10
The Fibonacci Series is as follows: 0 1 1 2 3 5 8 13 21 34
himani@Himani:~/Desktop/AI/fib$ ./fib_client localhost 12 13
Enter the number of series elements: 15
The Fibonacci Series is as follows: 0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
himani@Himani:~/Desktop/AI/fib$ ./fib_client localhost 12 13
Enter the number of series elements: 4
The Fibonacci Series is as follows: 0 1 1 2
himani@Himani:~/Desktop/AI/fib$
```

5. Find the maximum value of an array of integers using RPC.

Source Code for mval.x file

```
struct intpair {
    int a[200];
    int n;
};

program MVAL_PROG
{
    version MVAL_VERS
    {
        int MVAL(intpair) = 1;
    } = 1;

} = 0x23451111;
```

Source Code for mval_client.c file

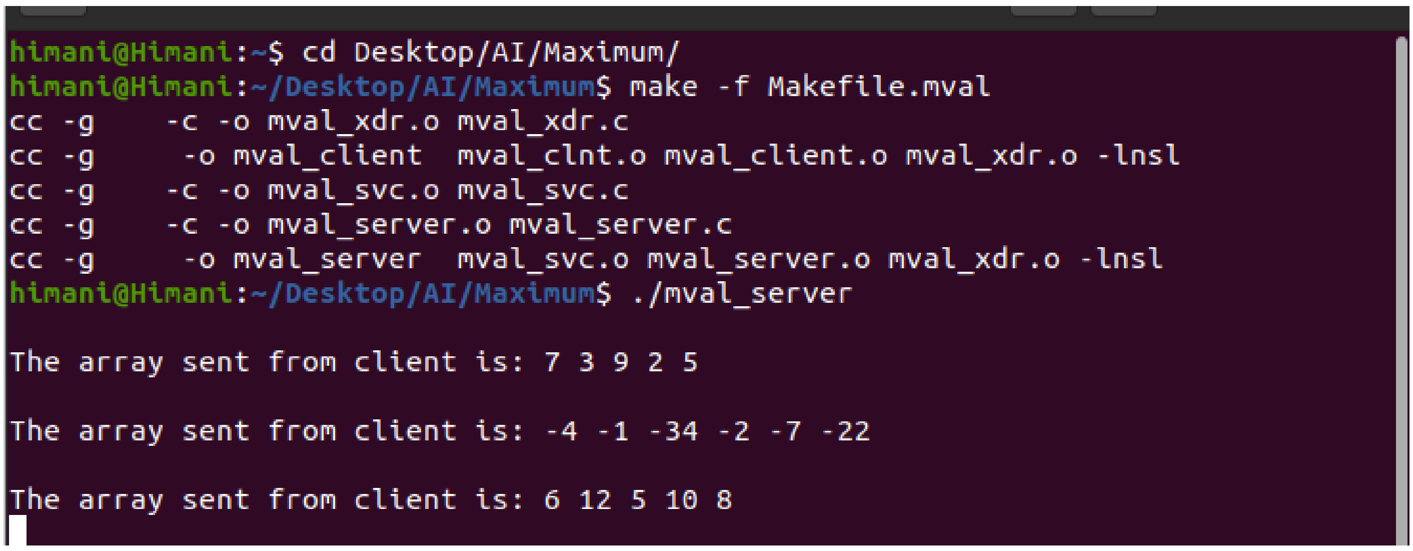
```
#include "mval.h"

void mval_prog_1(char *host,int len,int arr[])
{
    CLIENT *clnt;
    int  *result_1;
    intpair  mval_1_arg;

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

    mval_1_arg.n = len;
    int i = 0;
    for(i=0;i<len;i++){
        mval_1_arg.a[i] = arr[i];
    }

    result_1 = mval_1(&mval_1_arg, clnt);
    if (result_1 == (int *) NULL) {
        clnt_perror (clnt, "call failed");
    }else{
        printf("The maximum value is : %d\n",*result_1);
    }

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

int main (int argc, char *argv[])
{
    char *host;
    int n,i=0;
    if (argc < 2) {
        printf ("usage: %s server_host\n", argv[0]);
        exit (1);
    }
    host = argv[1];
    printf("Enter the number of array elements: ");
    scanf("%d",&n);
    int arr[n];
    printf("Enter the elements: ");
    for(i=0;i<n;i++){
        scanf("%d",&arr[i]);
    }
    mval_prog_1 (host,n,arr);
    exit (0);
}
```

Source Code for mval_server.c file

```
#include "mval.h"

int * mval_1_svc(intpair *argp, struct svc_req *rqstp)
{
    static int  result;
    int max = argp->a[0],i;
    int val = argp->n;

    printf("\nThe array sent from client is: ");
    for(i=0;i<val;i++){
        printf("%d ",argp->a[i]);
        if(argp->a[i]>max) max = argp->a[i];
    }
    printf("\n");

    result = max;
    return &result;
}
```

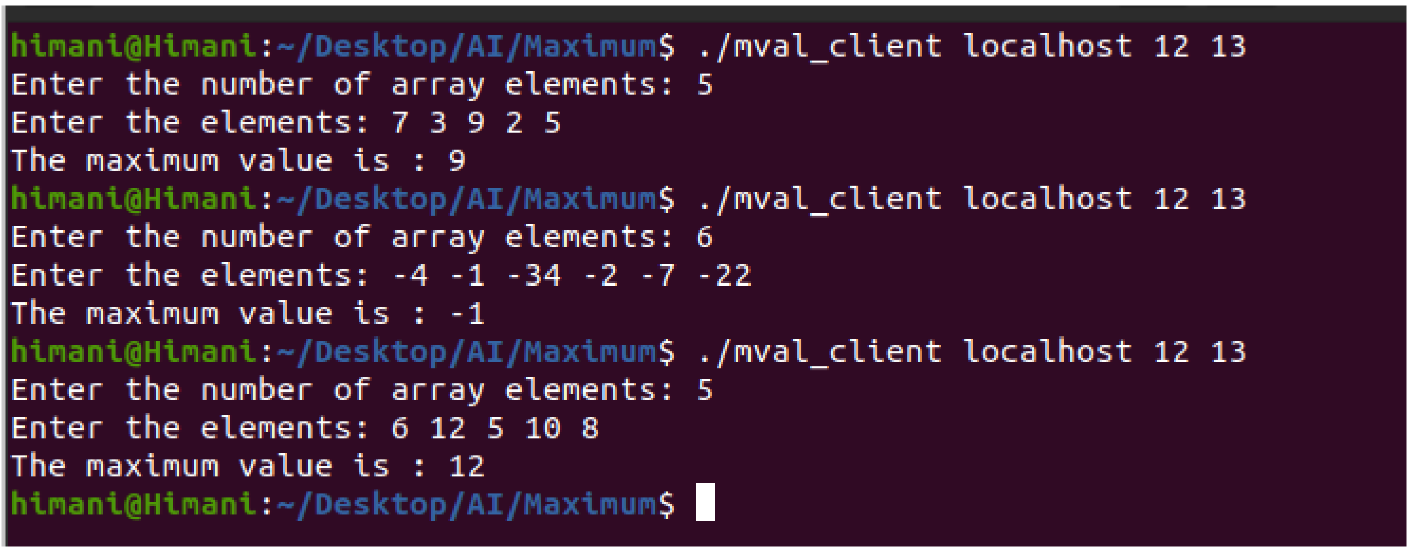
Output:

Server Side:



```
himani@Himani:~$ cd Desktop/AI/Maximum/
himani@Himani:~/Desktop/AI/Maximum$ make -f Makefile.mval
cc -g    -c -o mval_xdr.o mval_xdr.c
cc -g     -o mval_client  mval_clnt.o mval_client.o mval_xdr.o -lnsl
cc -g    -c -o mval_svc.o mval_svc.c
cc -g    -c -o mval_server.o mval_server.c
cc -g     -o mval_server  mval_svc.o mval_server.o mval_xdr.o -lnsl
himani@Himani:~/Desktop/AI/Maximum$ ./mval_server

The array sent from client is: 7 3 9 2 5

The array sent from client is: -4 -1 -34 -2 -7 -22

The array sent from client is: 6 12 5 10 8
```

Client Side:



```
himani@Himani:~/Desktop/AI/Maximum$ ./mval_client localhost 12 13
Enter the number of array elements: 5
Enter the elements: 7 3 9 2 5
The maximum value is : 9
himani@Himani:~/Desktop/AI/Maximum$ ./mval_client localhost 12 13
Enter the number of array elements: 6
Enter the elements: -4 -1 -34 -2 -7 -22
The maximum value is : -1
himani@Himani:~/Desktop/AI/Maximum$ ./mval_client localhost 12 13
Enter the number of array elements: 5
Enter the elements: 6 12 5 10 8
The maximum value is : 12
himani@Himani:~/Desktop/AI/Maximum$ 
```