U18CO018

DA ASSIGNMENT 5

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.

2. Implement Calculator (Basic operation).

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

4. Print out the Fibonacci series till the given number. 5. Find the maximum value of an array of integers using RPC.

A1)

fact.x struct
```
number {
       int n;
};

program FACT_PROG {
       version FACT_VERS {
       int fact(number)=1;
       }=1;
}=0x12345678;
```

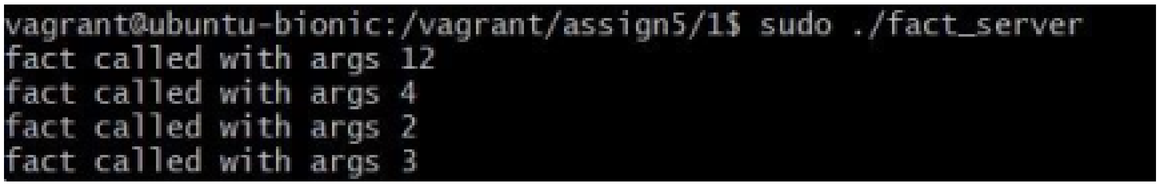
fact_server.c
```
#include "fact.h"

int *
fact_1_svc(number *argp, struct svc_req *rqstp)
{ static int result = 1;
       /*
        * insert server code here
        */ printf("fact called with args %d \n", argp-

       >n); int num = argp->n, ans = 1;
```

```
        for(int i = 1; i <= num; i++) {
                ans *= i;
        } result =

        ans;

        return &result;
}
```

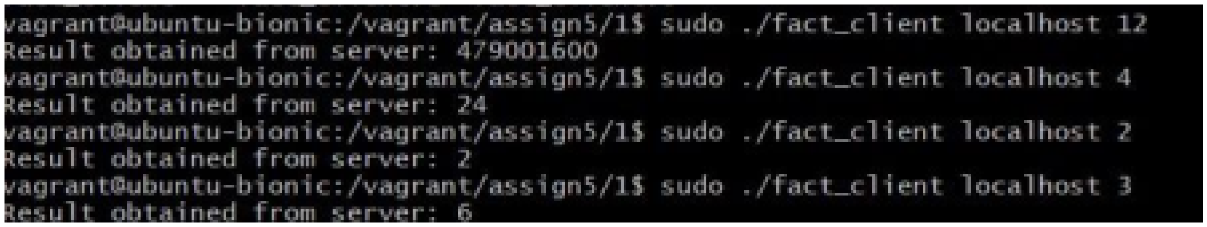
fact_client.c

```
#include "fact.h"

void fact_prog_1(char *host, int
num)
{
        CLIENT *clnt; int
        *result_1; number
        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.n = num;

        result_1 = fact_1(&fact_1_arg, clnt);
        if (result_1 == (int *) NULL) {
        clnt_perror (clnt, "call failed");
        } else { printf("Result obtained from server: %d\n",
                *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 NUMBER\n",
                argv[0]); exit (1);
        }
        host = argv[1]; fact_prog_1
        (host,atoi(argv[2]));
exit (0);
}
```

Output:



```
vagrant@ubuntu-bionic:/vagrant/assign5/1$ sudo ./fact_server
fact called with args 12
fact called with args 4
fact called with args 2
fact called with args 3
```



```
vagrant@ubuntu-bionic:/vagrant/assign5/1$ sudo ./fact_client localhost 12
Result obtained from server: 479001600
vagrant@ubuntu-bionic:/vagrant/assign5/1$ sudo ./fact_client localhost 4
Result obtained from server: 24
vagrant@ubuntu-bionic:/vagrant/assign5/1$ sudo ./fact_client localhost 2
Result obtained from server: 2
vagrant@ubuntu-bionic:/vagrant/assign5/1$ sudo ./fact_client localhost 3
Result obtained from server: 6
```

A2)

calculate.x

```
struct inputs{
   float num1;
   float num2;
    char op;
};

program CALCULATE_PROG{
    version
        CALCULATE_VERS{ float
        add(inputs)=1; float
        sub(inputs)=2; float
        mul(inputs)=3; float
        div(inputs)=4;
    }=1;
}=0x12345678;
```

calculate_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 "calculate.h"


void calculate_prog_1(char *host, int num1, int num2,
char op)
```

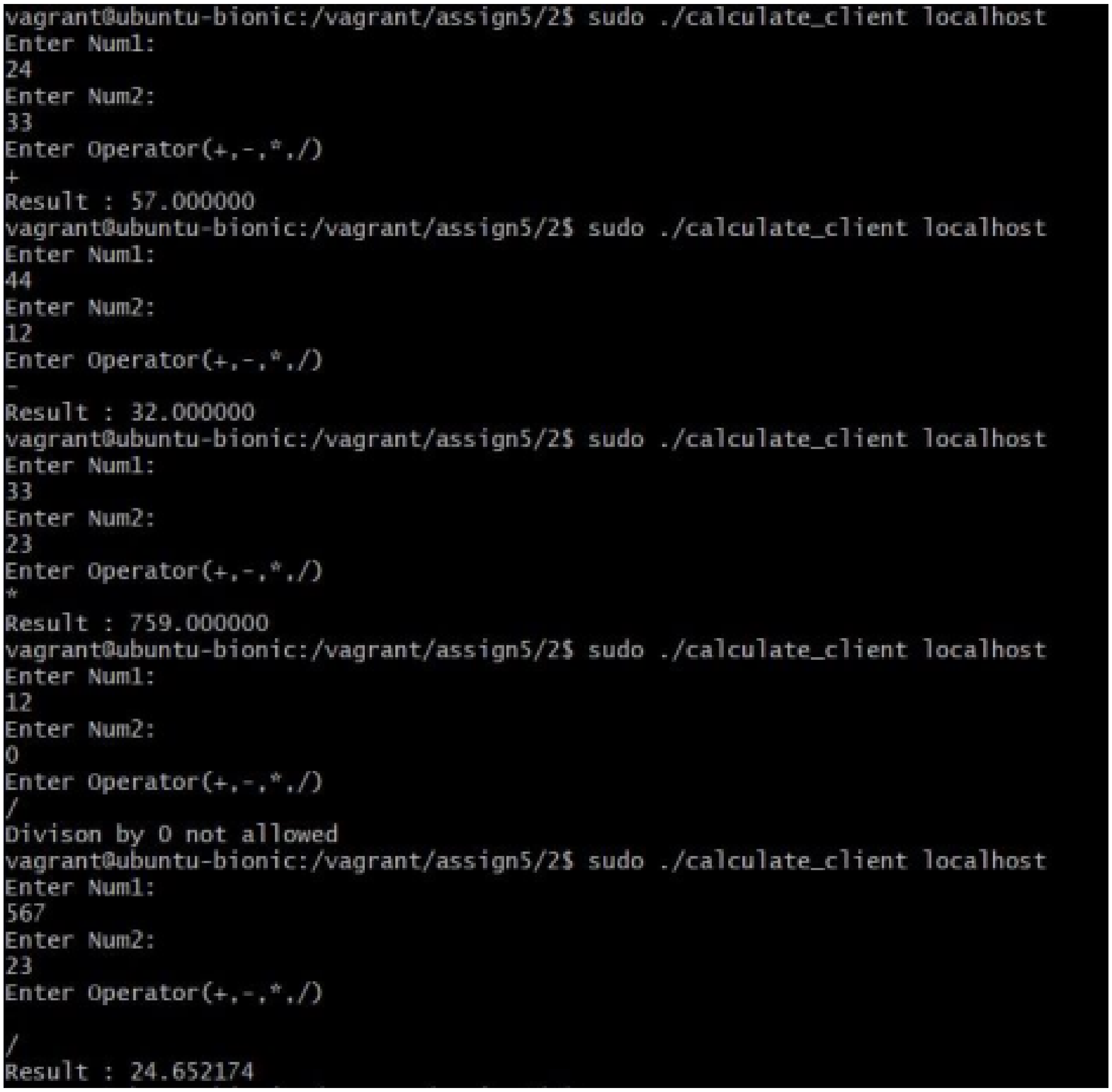
```
{
          CLIENT *clnt; float
          *result_1; inputs
          add_1_arg; float
          *result_2; inputs
          sub_1_arg; float
          *result_3; inputs
          mul_1_arg; float
          *result_4; inputs
          div_1_arg;

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

          if(op == '+') {
                    add_1_arg.num1=num1;
                    add_1_arg.num2=num2;
                    add_1_arg.op = op; result_1 =
                    add_1(&add_1_arg, clnt); if
                    (result_1 == (float *) NULL) {
                    clnt_perror (clnt, "call failed");
                    } else { printf("Result : %f \n",
                    *result_1);
                    } } else
          if(op == '-') {
                    sub_1_arg.num1=num1;
                    sub_1_arg.num2=num2;
                    sub_1_arg.op = op; result_2 =
                    sub_1(&sub_1_arg, clnt); if
                    (result_2 == (float *) NULL) {
                    clnt_perror (clnt, "call failed");
                    } else { printf("Result : %f \n",
                    *result_2);
                    } } else
          if(op == '*') {
                    mul_1_arg.num1=num1;
                    mul_1_arg.num2=num2;
                    mul_1_arg.op = op; result_3 =
                    mul_1(&mul_1_arg, clnt); if
                    (result_3 == (float *) NULL) {
                    clnt_perror (clnt, "call failed");
                    } else { printf("Result : %f \n",
                    *result_3);
                    }
          }
          else
```

```
        {
                div_1_arg.num1=num1;
                div_1_arg.num2=num2; div_1_arg.op =
                op; if(num2 == 0) { printf("Divison by 0 not
                allowed\n");
                        exit(0);
                }
                result_4 = div_1(&div_1_arg, clnt); if
                (result_4 == (float *) NULL) {
                clnt_perror (clnt, "call failed");
                } else { printf("Result : %f \n",
                *result_4);
                }
        }

#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];



        float a, b; char op; printf("Enter
        Num1: \n"); scanf("%f",&a);
        printf("Enter Num2: \n");
        scanf("%f",&b); printf("Enter
        Operator(+,-,*,/)\n");
        scanf("%s",&op);

        calculate_prog_1 (host, a, b, op);
exit (0);
}
```

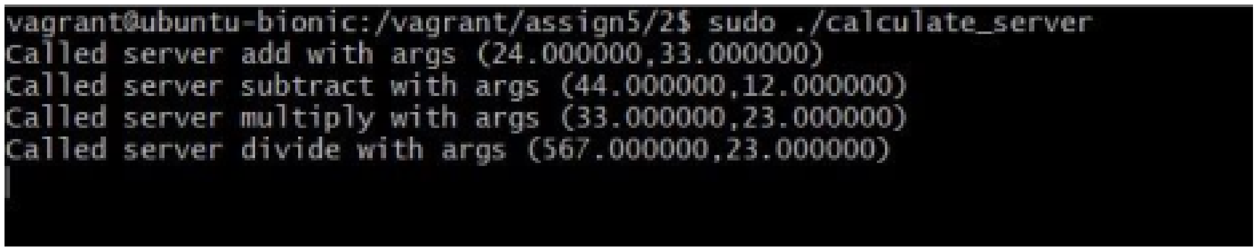
calculate_server.c

```
/*
         * 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 "calculate.h"

float * add_1_svc(inputs *argp, struct svc_req
*rqstp)
{ static float result;

        /*
         * insert server code here
         */
        printf("Called server add with args (%f,%f)\n", argp->num1, argp->num2);

        result = argp->num1 + argp->num2;


        return &result;
}

float * sub_1_svc(inputs *argp, struct svc_req
*rqstp)
{ static float result;

        /*
         * insert server code here
         */

        printf("Called server subtract with args (%f,%f)\n", argp->num1,

        argp- >num2); result = argp->num1 - argp->num2;


        return &result;
}

float * mul_1_svc(inputs *argp, struct svc_req
*rqstp)
{ static float result;
        /*
         * insert server code here
         */
        printf("Called server multiply with args (%f,%f)\n", argp->num1, argp->num2);

        result = argp->num1 * argp->num2;


        return &result;
}

float * div_1_svc(inputs *argp, struct svc_req
*rqstp)
{ static float result;

	/*
	 * insert server code here
	 */
	printf("Called server divide with args (%f,%f)\n", argp->num1, argp->num2);

	result = argp->num1 / argp->num2;

	return &result;
}
```



```
vagrant@ubuntu-bionic:/vagrant/assign5/2$ sudo ./calculate_client localhost
Enter Num1:
24
Enter Num2:
33
Enter Operator(+,-,*,/)
+
Result : 57.000000
vagrant@ubuntu-bionic:/vagrant/assign5/2$ sudo ./calculate_client localhost
Enter Num1:
44
Enter Num2:
12
Enter Operator(+,-,*,/)
-
Result : 32.000000
vagrant@ubuntu-bionic:/vagrant/assign5/2$ sudo ./calculate_client localhost
Enter Num1:
33
Enter Num2:
23
Enter Operator(+,-,*,/)
*
Result : 759.000000
vagrant@ubuntu-bionic:/vagrant/assign5/2$ sudo ./calculate_client localhost
Enter Num1:
12
Enter Num2:
0
Enter Operator(+,-,*,/)
/
Divison by 0 not allowed
vagrant@ubuntu-bionic:/vagrant/assign5/2$ sudo ./calculate_client localhost
Enter Num1:
567
Enter Num2:
23
Enter Operator(+,-,*,/)

/
Result : 24.652174
```



```
vagrant@ubuntu-bionic:/vagrant/assign5/2$ sudo ./calculate_server
Called server add with args (24.000000,33.000000)
Called server subtract with args (44.000000,12.000000)
Called server multiply with args (33.000000,23.000000)
Called server divide with args (567.000000,23.000000)
```

A3)

prime.x

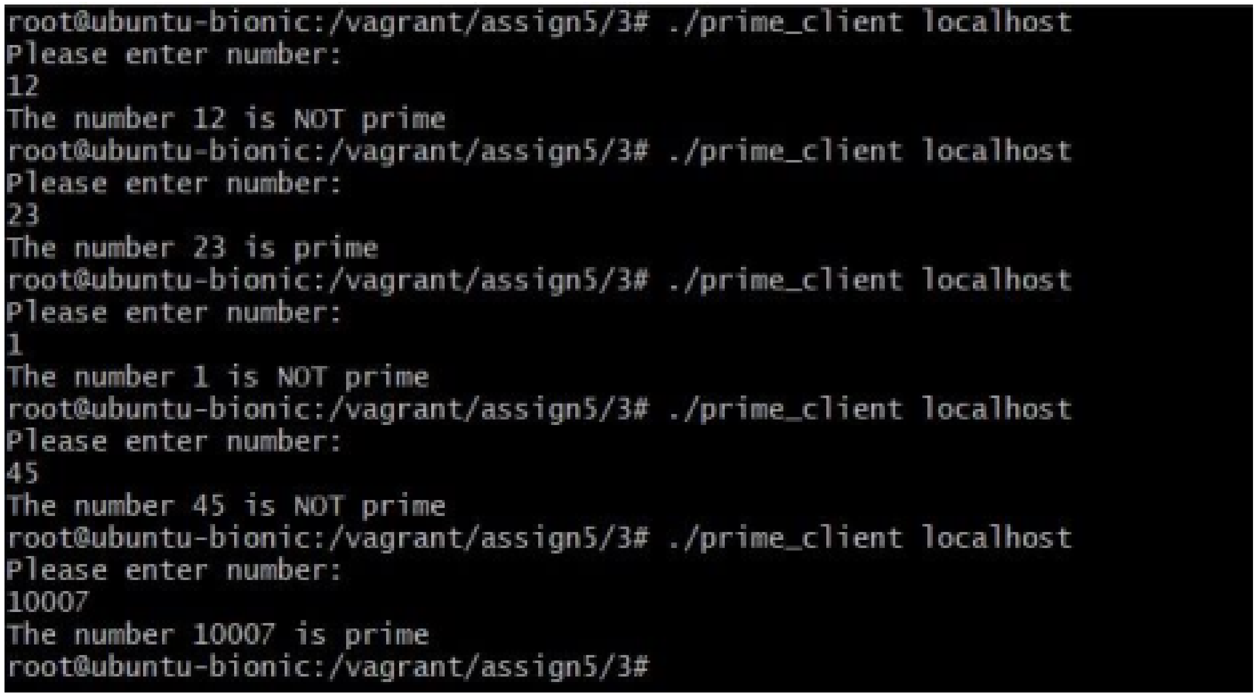
```
struct input{ int num;
};

program PRIME_PROG{
   version PRIME_VERS{
       bool isprime(input)=1;
   }=1;
}=0x12345678;
```

prime_client.c

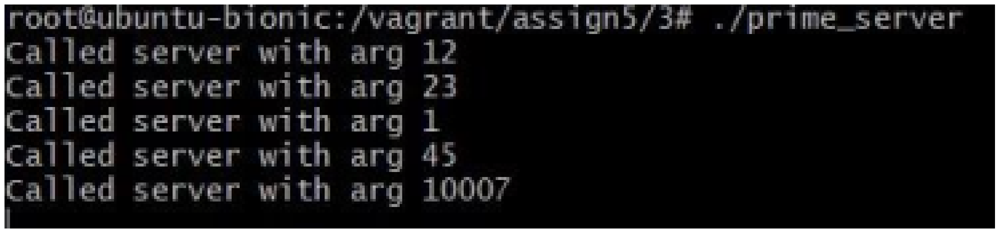
```
/*
 * This is sample code generated by rpcgen.
 * These are only templates and you can use them*
   as a guideline for developing your own functions.
*/

#include "prime.h"


void prime_prog_1(char *host, int
 num)
{
        CLIENT *clnt; bool_t
        *result_1; input
        isprime_1_arg;

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

        isprime_1_arg.num = num; result_1 =
        isprime_1(&isprime_1_arg, clnt); if (result_1
        == (bool_t *) NULL) { clnt_perror (clnt, "call
        failed");
        } else { if(*result_1) { printf("The number %d is prime\n",
                    num);
                    } else { printf("The number %d is NOT prime\n",
                              num);
                    }
          }
#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]; printf("Please
          enter number: \n"); int num;
          scanf("%d",&num);
          prime_prog_1 (host, num);
exit (0);
}
```

prime_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 "prime.h"
#include <stdbool.h>

bool_t * isprime_1_svc(input *argp, struct
svc_req *rqstp)
{ static bool_t result;

          /*
* insert server code here
           */

          printf("Called server with arg %d\n",
           argp->num); bool isPrime = true; int num =
                  argp->num; if(num == 1)
          isPrime = false; for(int i = 2; i *
           i <= num; i++) { if(num % i ==
                    0) { isPrime = false;
                    }
            }
            result = isPrime;
            return &result;
}
```



```
root@ubuntu-bionic:/vagrant/assign5/3# ./prime_client localhost
Please enter number:
12
The number 12 is NOT prime
root@ubuntu-bionic:/vagrant/assign5/3# ./prime_client localhost
Please enter number:
23
The number 23 is prime
root@ubuntu-bionic:/vagrant/assign5/3# ./prime_client localhost
Please enter number:
1
The number 1 is NOT prime
root@ubuntu-bionic:/vagrant/assign5/3# ./prime_client localhost
Please enter number:
45
The number 45 is NOT prime
root@ubuntu-bionic:/vagrant/assign5/3# ./prime_client localhost
Please enter number:
10007
The number 10007 is prime
root@ubuntu-bionic:/vagrant/assign5/3#
```



```
root@ubuntu-bionic:/vagrant/assign5/3# ./prime_server
Called server with arg 12
Called server with arg 23
Called server with arg 1
Called server with arg 45
Called server with arg 10007
```

A4)

fib.x

```
struct
 input{ int
 num;
};

program FIB_PROG{
   version FIB_VERS{
       string fib(input)=1;
    }=1;
}=0x12345678;
```

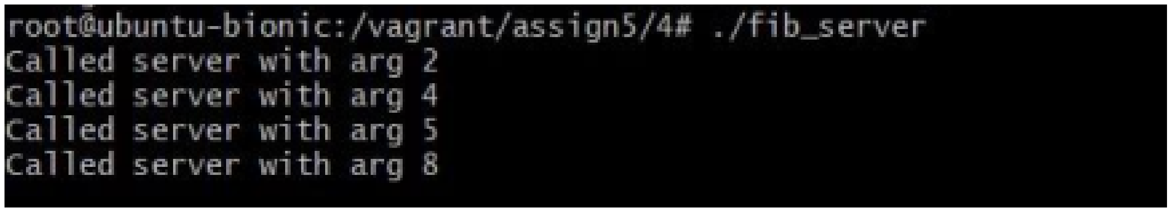
fib_client.c

```
/*
 * This is sample code generated by rpcgen.
 * These are only templates and you can use them*
```

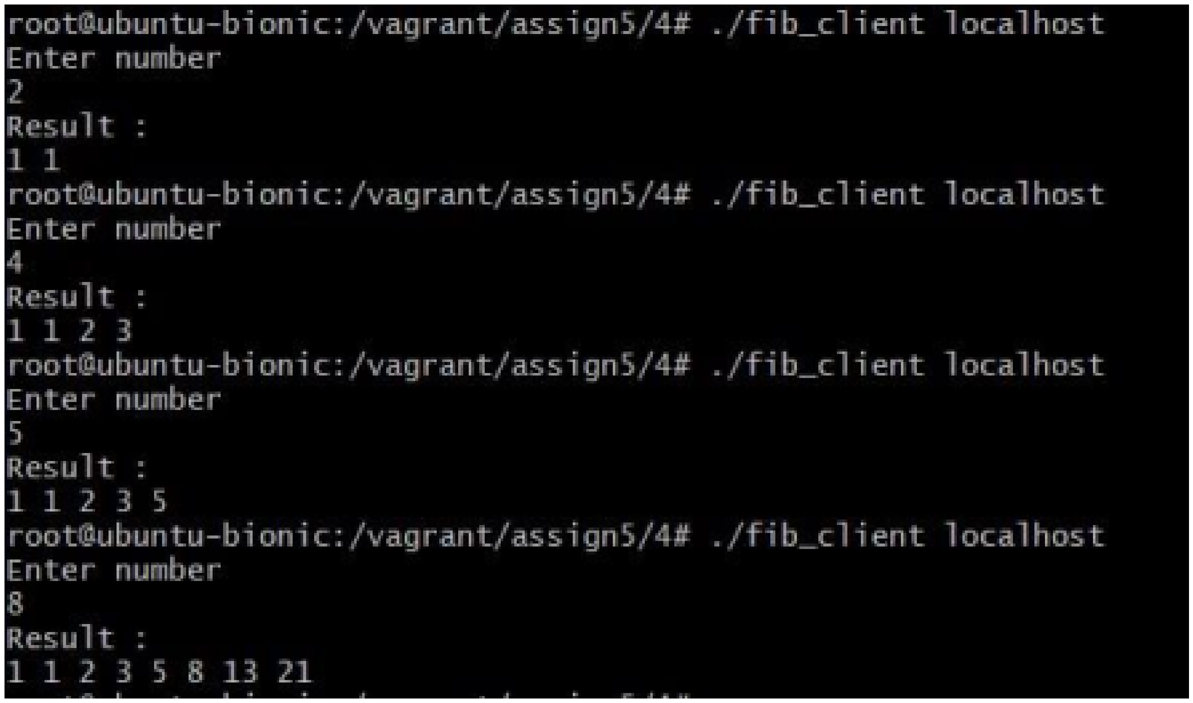
```
as a guideline for developing your own functions.
*/

#include "fib.h"


void
fib_prog_1(char *host, int n)
{
        CLIENT *clnt;
        char * *result_1;
        input 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.num=n; result_1 =
        fib_1(&fib_1_arg, clnt); if (result_1
        == (char **) NULL) { clnt_perror
        (clnt, "call failed");
        } else { if(n == 0) { printf("enter number >=
                1\n"); exit(0);
                } printf("Result : \n");
                printf("%s\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]; int n;
        printf("Enter number
        \n"); scanf("%d", &n);
        fib_prog_1 (host,n);
exit (0);
}
```

fib_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 "fib.h"

char ** fib_1_svc(input *argp, struct svc_req
*rqstp)
{ static char * result;

        /*
* insert server code here
         */
        printf("Called server with arg %d\n", argp-
        >num); int num = argp->num; if(num == 1)
                { result = "1";
        } else if(num == 2) {
                result = "1 1";
        } else { int t1 = 0, t2 = 1; int next
                = 1; int a[num+1]; a[0] =
                0, a[1] = 1, a[2] = 1; char
                str[10000]; int index = 0;
                for (int i=3; i<=num; i++) {
                a[i] = a[i-1] + a[i-2];
                } for(int i = 1; i <= num; i++) { index +=
                sprintf(&str[index], "%d ", a[i]);
                } result =
                str;
        } return
        &result;
}
```



```
root@ubuntu-bionic:/vagrant/assign5/4# ./fib_server
Called server with arg 2
Called server with arg 4
Called server with arg 5
Called server with arg 8
```



```
root@ubuntu-bionic:/vagrant/assign5/4# ./fib_client localhost
Enter number
2
Result :
1 1
root@ubuntu-bionic:/vagrant/assign5/4# ./fib_client localhost
Enter number
4
Result :
1 1 2 3
root@ubuntu-bionic:/vagrant/assign5/4# ./fib_client localhost
Enter number
5
Result :
1 1 2 3 5
root@ubuntu-bionic:/vagrant/assign5/4# ./fib_client localhost
Enter number
8
Result :
1 1 2 3 5 8 13 21
```

A5)

max.x

```
struct input{
   int n; int
   arr[100];
};

program FIB_PROG{
   version FIB_VERS{
   int fib(input)=1;
   }=1;
}=0x12345678;
```

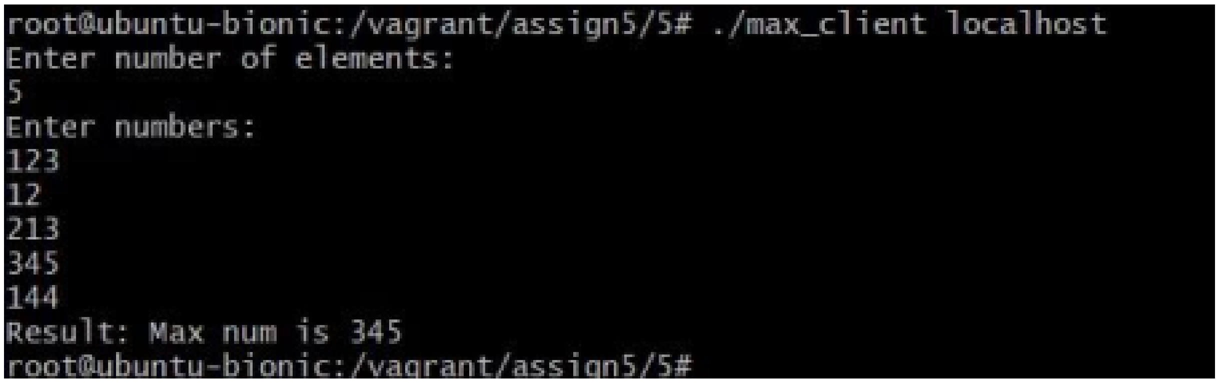
max_client.c

```
/*
* 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 "max.h"
void fib_prog_1(char *host, int n, int
*arr)
{
   CLIENT *clnt;
   int *result_1;
   input fib_1_arg;

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

   int a[n];
   fib_1_arg.n = n;
   // fib_1_arg.arr = (int*)malloc(sizeof(int) * n);

   for(int i = 0; i < n; i++) {
       fib_1_arg.arr[i] = arr[i];
   }

   result_1 = fib_1(&fib_1_arg, clnt);
   if (result_1 == (int *) NULL) {
       clnt_perror (clnt, "call failed");
   } else { printf("Result: Max num 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]; int n; printf("Enter
   number of elements: \n"); int arr[100];
   scanf("%d",&n); printf("Enter
   numbers: \n"); for(int i = 0; i < n; i++) {
       scanf("%d", &arr[i]);
   }
   fib_prog_1 (host,n,arr);
exit (0);
}
```

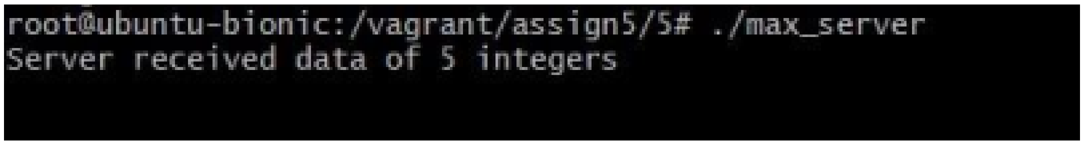
max_server.c

```
/*
* 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 "max.h"

int * fib_1_svc(input *argp, struct svc_req
*rqstp)
{ static int result;

    /*
* insert server code here
     */ int *arr = argp->arr; int n = argp->n; int mx =
    arr[0]; printf("Server received data of %d
    integers\n", n); for(int i = 0; i < n; i++) {
        if(mx < arr[i]) mx = arr[i];
    }
    result = mx;
    return &result;
}
```



```
root@ubuntu-bionic:/vagrant/assign5/5# ./max_client localhost
Enter number of elements:
5
Enter numbers:
123
12
213
345
144
Result: Max num is 345
root@ubuntu-bionic:/vagrant/assign5/5#
```



```
root@ubuntu-bionic:/vagrant/assign5/5# ./max_server
Server received data of 5 integers
```