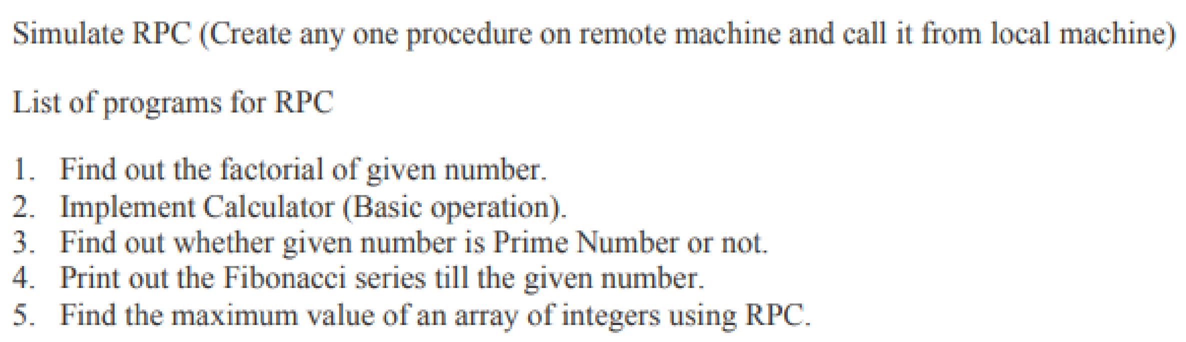
DA LAB 5

SAHIL BONDRE: U18CO021



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.

Q1:

fact.x

```
struct intpair {
    int a;
};


program FACT_PROG {
    version FACT_VERS {
        int FACT(intpair) = 1;
    } = 1;
} = 0x23451111;
```

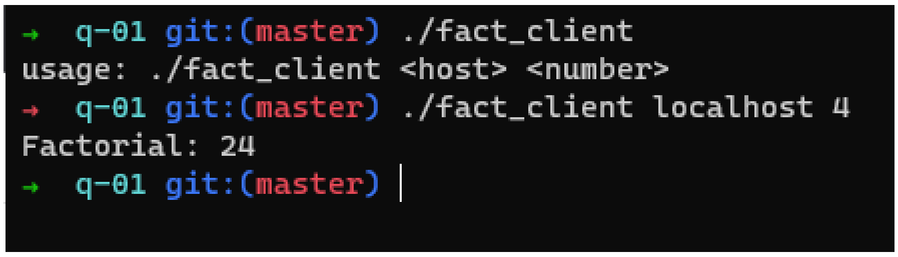
fact_client.c

```
#include "fact.h"


void fact_prog_1(char *host, int a) {
```

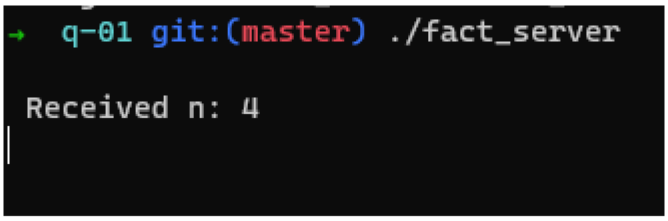
```c
  CLIENT *clnt;
  int *result_1;
  intpair fact_1_arg;
  clnt = clnt_create(host, FACT_PROG, FACT_VERS, "udp");
  if (clnt == NULL) {
    clnt_pcreateerror(host);
    exit(1);
  }
  fact_1_arg.a = a;
  result_1 = fact_1(&fact_1_arg, clnt);
  if (result_1 == (int *)NULL) {
    clnt_perror(clnt, "call failed:");
  } else {
    printf("Factorial: %d\n", *result_1);
  }
  clnt_destroy(clnt);
}

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


  if (argc < 3) {
    printf("usage: %s <host> <number>\n", argv[0]);
    exit(1);
  }
  host = argv[1];
  a = atoi(argv[2]);
  fact_prog_1(host, a);
  exit(0);
}
```

fact_server.c

```
#include "fact.h"

int *fact_1_svc(intpair *argp, struct svc_req *rqstp) {
  static int result, n, fact;
  int i;
  n = argp->a;
  // factorial logic
  fact = 1;
  printf("\n Received n: %d \n", n);
  for (i = n; i > 0; i--) {
    fact = fact * i;
  }
  result = fact;
  return &result;
}
```



```
→  q-01 git:(master) ./fact_client
usage: ./fact_client <host> <number>
→  q-01 git:(master) ./fact_client localhost 4
Factorial: 24
→  q-01 git:(master)
```



```
→  q-01 git:(master) ./fact_server

 Received n: 4
```

Q2:

calculate.x

```
struct inputs{

    float num1;

    float num2;

    char operator;

};


program CALCULATE_PROG{

    version CALCULATE_VER{

        float ADD(inputs)=1;

        float SUB(inputs)=2;

        float MUL(inputs)=3;

        float DIV(inputs)=4;

    }=1;

}=0x2fffffff;
```

calculate_client.c

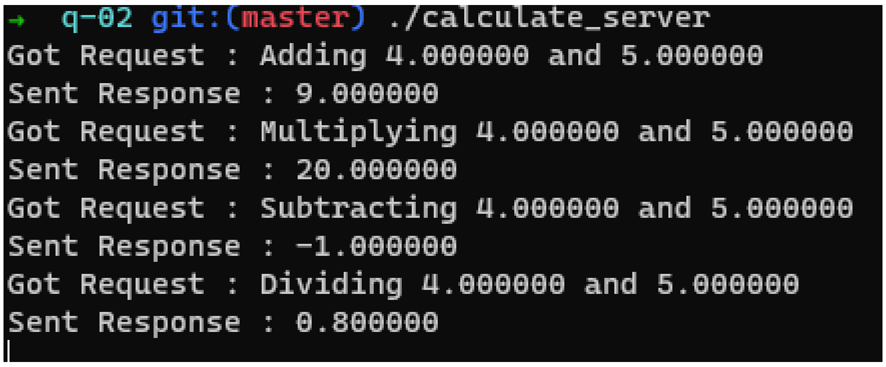
```
/*

 * 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 a, int b, char c) {

  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;



  clnt = clnt_create(host, CALCULATE_PROG, CALCULATE_VER, "tcp");

  if (clnt == NULL) {

    clnt_pcreateerror(host);

    exit(1);
```

```
}


if (c == '+') {

  add_1_arg.num1 = a;

  add_1_arg.num2 = b;

  result_1 = add_1(&add_1_arg, clnt);

  if (result_1 == (float *)NULL) {

    clnt_perror(clnt, "call failed");

  }

  printf("%f\n", *result_1);

} else if (c == '-') {

  sub_1_arg.num1 = a;

  sub_1_arg.num2 = b;

  result_2 = sub_1(&sub_1_arg, clnt);

  if (result_2 == (float *)NULL) {

    clnt_perror(clnt, "call failed");

  }

  printf("%f\n", *result_2);

} else if (c == 'x') {

  mul_1_arg.num1 = a;

  mul_1_arg.num2 = b;

  result_3 = mul_1(&mul_1_arg, clnt);

  if (result_3 == (float *)NULL) {

    clnt_perror(clnt, "call failed");

  }

  printf("%f\n", *result_3);
  } else {

    div_1_arg.num1 = a;

    div_1_arg.num2 = b;

    if (b == 0) {

      printf("Error: Cannot divide by zero\n");

    } else {

      result_4 = div_1(&div_1_arg, clnt);

      if (result_4 == (float *)NULL) {

        clnt_perror(clnt, "call failed");

      }

      printf("%f\n", *result_4);

    }

  }

  clnt_destroy(clnt);

}


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

  char *host;


  if (argc < 5) {

    printf("usage: %s <host> <num 1> <op> <num 2>\n", argv[0]);

    exit(1);

  }

  host = argv[1];

  float num1 = atof(argv[2]);

  char op = argv[3][0];
  float num2 = atof(argv[4]);


  calculate_prog_1(host, num1, num2, op);

  exit(0);

}
```

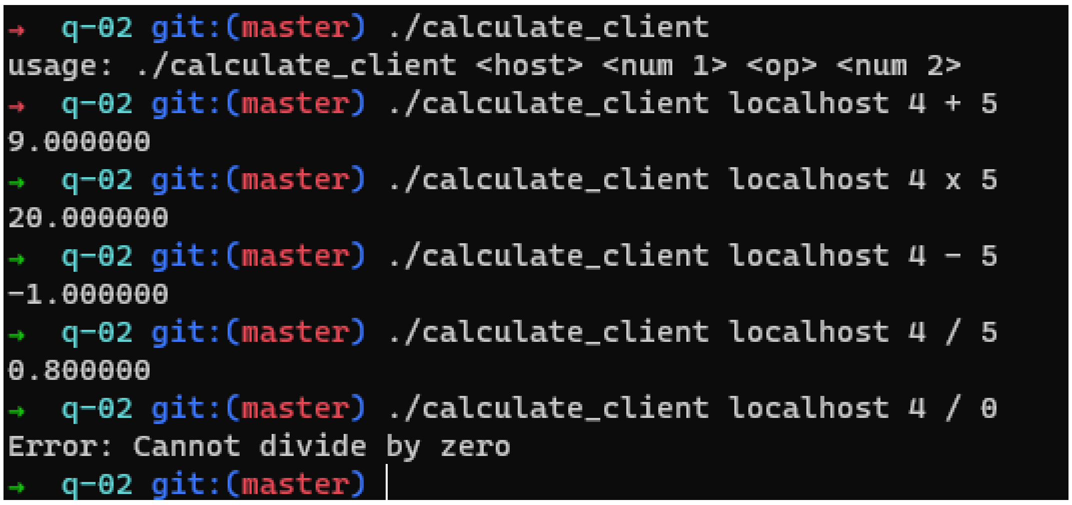
calculate_server.c

```
/*

 * 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"

#include<stdlib.h>

#include<stdio.h>


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

  static float result;

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

  printf("Got Request : Adding %f and %f\n", argp->num1, argp->num2);

  printf("Sent Response : %f\n", result);

  return (&result);

}
```

```c
float *sub_1_svc(inputs *argp, struct svc_req *rqstp) {

  static float result;

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

  printf("Got Request : Subtracting %f and %f\n", argp->num1, argp->num2);

  printf("Sent Response : %f\n", result);

  return (&result);

}


float *mul_1_svc(inputs *argp, struct svc_req *rqstp) {

  static float result;

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

  printf("Got Request : Multiplying %f and %f\n", argp->num1, argp->num2);

  printf("Sent Response : %f\n", result);

  return (&result);

}


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

  static float result;

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

  printf("Got Request : Dividing %f and %f\n", argp->num1, argp->num2);

  printf("Sent Response : %f\n", result);

  return (&result);

}
```



```
→  q-02 git:(master) ./calculate_server
Got Request : Adding 4.000000 and 5.000000
Sent Response : 9.000000
Got Request : Multiplying 4.000000 and 5.000000
Sent Response : 20.000000
Got Request : Subtracting 4.000000 and 5.000000
Sent Response : -1.000000
Got Request : Dividing 4.000000 and 5.000000
Sent Response : 0.800000
```



```
→  q-02 git:(master) ./calculate_client
usage: ./calculate_client <host> <num 1> <op> <num 2>
→  q-02 git:(master) ./calculate_client localhost 4 + 5
9.000000
→  q-02 git:(master) ./calculate_client localhost 4 x 5
20.000000
→  q-02 git:(master) ./calculate_client localhost 4 - 5
-1.000000
→  q-02 git:(master) ./calculate_client localhost 4 / 5
0.800000
→  q-02 git:(master) ./calculate_client localhost 4 / 0
Error: Cannot divide by zero
→  q-02 git:(master)
```

Q3:

prime.x

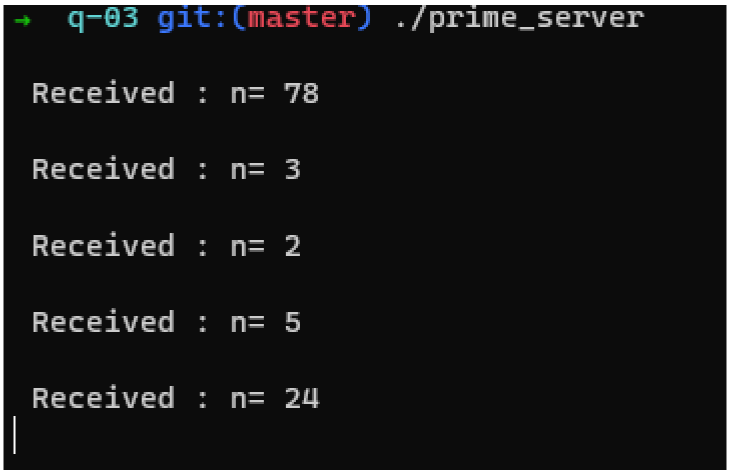
```
struct intpair {

    int a;

};


program PRIME_PROG {

    version PRIME_VERS {

        int PRIME(intpair) = 1;

    } = 1;

} = 0x23451111;
```

prime_client.c

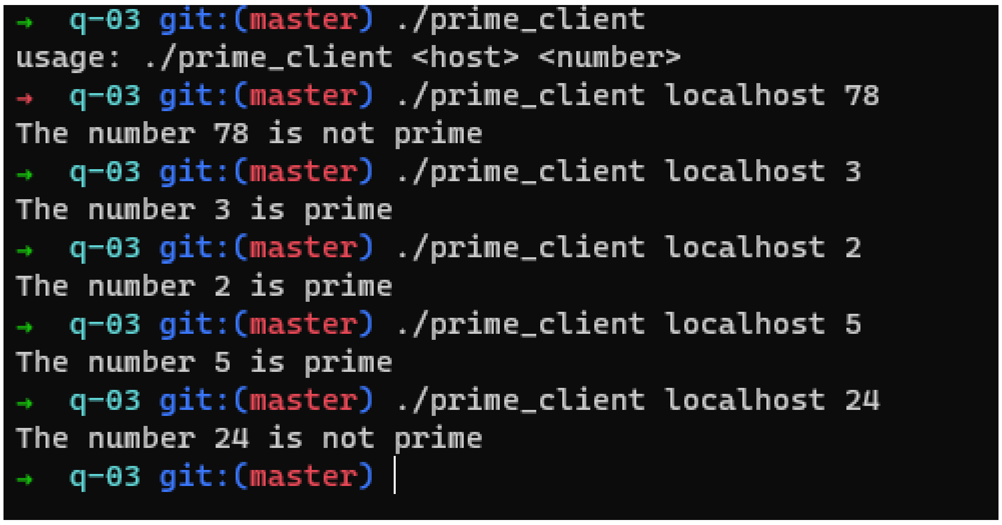
```c
/*
 * This is sample code generated by rpcgen.
 * These are only templates and you can use them
 * as a guideline for developing your own functions.
 */

#include "prime.h"

void prime_prog_1(char *host, int a) {
  CLIENT *clnt;
  int *result_1;
  intpair prime_1_arg;
  clnt = clnt_create(host, PRIME_PROG, PRIME_VERS, "udp");
  if (clnt == NULL) {
    clnt_pcreateerror(host);
    exit(1);
  }
  prime_1_arg.a = a;
  result_1 = prime_1(&prime_1_arg, clnt);
  if (result_1 == (int *)NULL) {
    clnt_perror(clnt, "call failed:");
  } else {
    if (*result_1 == 1) {
      printf("The number %d is not prime\n", a);
    } else {
      printf("The number %d is prime\n", a);
    }
  }
  clnt_destroy(clnt);
}

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

  if (argc < 3) {
    printf("usage: %s <host> <number>\n", argv[0]);
    exit(1);
  }
  host = argv[1];
  int number = atoi(argv[2]);
  prime_prog_1(host, number);
  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(intpair *argp, struct svc_req *rqstp) {
  static int result;
  int n = argp->a, flag = 0, i;
  printf("\n Received : n= %d \n", n);
  if (n == 0 || n == 1) flag = 1;
  for (i = 2; i <= n / 2; ++i) {
    if (n % i == 0) {
      flag = 1;
      break;
    }
  }
  if (!flag)
    result = 0;
  else
    result = 1;
  return (&result);
}
```



```
→  q-03 git:(master) ./prime_server

 Received : n= 78

 Received : n= 3

 Received : n= 2

 Received : n= 5

 Received : n= 24
```



```
→  q-03 git:(master) ./prime_client
usage: ./prime_client <host> <number>
→  q-03 git:(master) ./prime_client localhost 78
The number 78 is not prime
→  q-03 git:(master) ./prime_client localhost 3
The number 3 is prime
→  q-03 git:(master) ./prime_client localhost 2
The number 2 is prime
→  q-03 git:(master) ./prime_client localhost 5
The number 5 is prime
→  q-03 git:(master) ./prime_client localhost 24
The number 24 is not prime
→  q-03 git:(master)
```

Q4:

fibonacci.x

```
struct intpair {

    int a;

    };

program FIBONACCI_PROG {

    version FIBONACCI__VERS {

        int FIBONACCI_(intpair) = 1;

    } = 1;

} = 0x23451111;
```

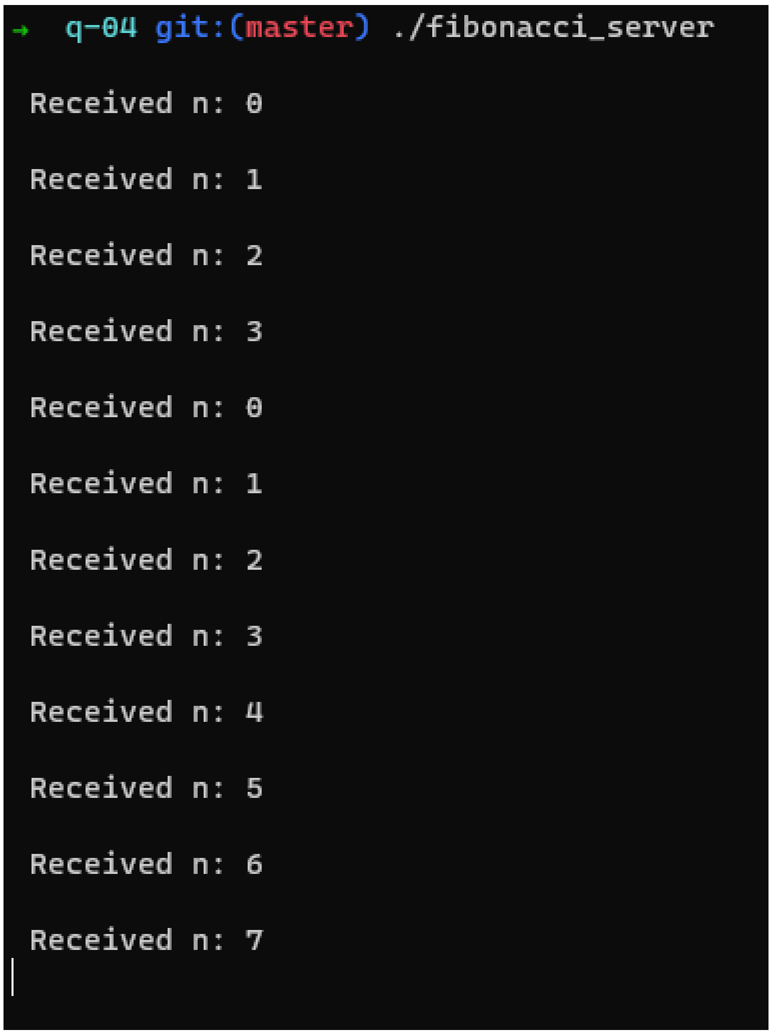
fibonacci_client.c

```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 "fibonacci.h"



void fibonacci_prog_1(char *host, int a) {

  CLIENT *clnt;

  int *result_1;

  intpair fibonacci__1_arg;

  clnt = clnt_create(host, FIBONACCI_PROG, FIBONACCI__VERS, "udp");

  if (clnt == NULL) {

    clnt_pcreateerror(host);

    exit(1);

  }

  for (int i = 0; i < a; ++i) {

    fibonacci__1_arg.a = i;

    result_1 = fibonacci__1(&fibonacci__1_arg, clnt);

    if (result_1 == (int *)NULL) {

      clnt_perror(clnt, "call failed:");

    } else {
      printf("%d ", *result_1);

    }

  }

  printf("\n");

  clnt_destroy(clnt);

}


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

  char *host;

  int number;


  if (argc < 3) {

    printf("usage: %s <host> <n>\n", argv[0]);

    exit(1);

  }

  host = argv[1];

  number = atoi(argv[2]);

  fibonacci_prog_1(host, number);

  exit(0);

}
```

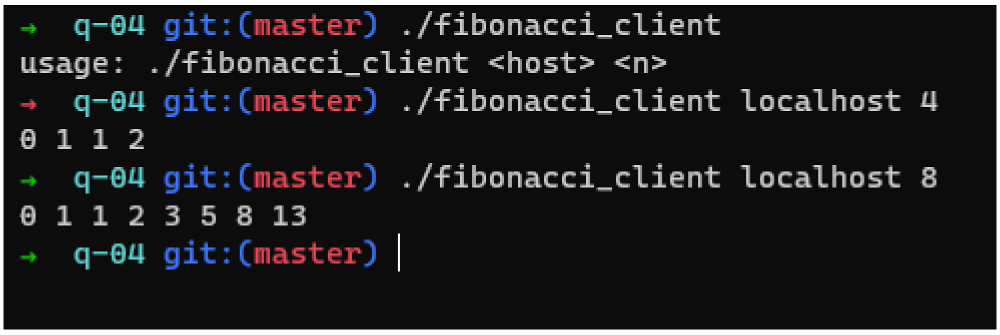
fibonacci_server.c

```
/*

 * This is sample code generated by rpcgen.
```

```c
 * These are only templates and you can use them
 * as a guideline for developing your own functions.
 */

#include <stdio.h>
#include <stdlib.h>

#include "fibonacci.h"
int calculate_fib(int n) {
  if (n <= 1) return n;
  return calculate_fib(n - 1) + calculate_fib(n - 2);
}

int *fibonacci__1_svc(intpair *argp, struct svc_req *rqstp) {
  int n = argp->a;
  static int result;
  printf("\n Received n: %d \n", n);
  result = calculate_fib(n);
  return (&result);
}
```



```
➜  q-04 git:(master) ./fibonacci_server

 Received n: 0

 Received n: 1

 Received n: 2

 Received n: 3

 Received n: 0

 Received n: 1

 Received n: 2

 Received n: 3

 Received n: 4

 Received n: 5

 Received n: 6

 Received n: 7
```



```
➜  q-04 git:(master) ./fibonacci_client
usage: ./fibonacci_client <host> <n>
➜  q-04 git:(master) ./fibonacci_client localhost 4
0 1 1 2
➜  q-04 git:(master) ./fibonacci_client localhost 8
0 1 1 2 3 5 8 13
➜  q-04 git:(master) 
```

Q5:

max.x

```
struct intarr {

    int a;

    int b;
};

program MAX_PROG {

    version MAX__VERS {

        int MAX_(intarr) = 1;

    } = 1;

} = 0x23451111;

```

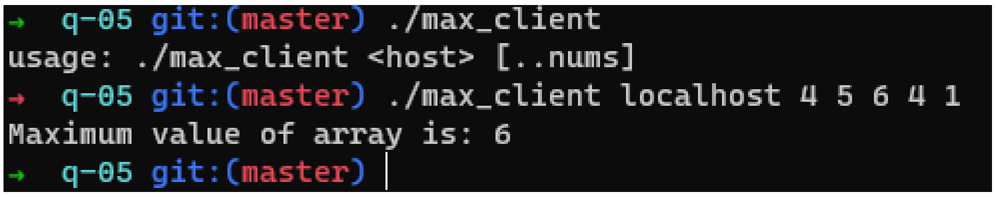
max_client.c

```
/*

 * 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 max_prog_1(char *host, int *a, int n) {

  CLIENT *clnt;

  int *result_1;

  intarr max__1_arg;

  clnt = clnt_create(host, MAX_PROG, MAX__VERS, "udp");

  if (clnt == NULL) {

    clnt_pcreateerror(host);

    exit(1);

  }
```

```c
  max__1_arg.a = a[0];
  for (int i = 1; i < n; ++i) {
    max__1_arg.b = a[i];
    result_1 = max__1(&max__1_arg, clnt);
    if (result_1 == (int *)NULL) {
      clnt_perror(clnt, "call failed:");
    } else {
      if (*result_1) max__1_arg.a = a[i];
    }
  }
  printf("Maximum value of array is: %d\n", max__1_arg.a);
  clnt_destroy(clnt);
}

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

  if (argc < 3) {
    printf("usage: %s <host> [..nums]\n", argv[0]);
    exit(1);
  }
  host = argv[1];
  int *nums = (int *)malloc(sizeof(int) * (argc - 2));
  for (int i = 2; i < argc; ++i) {
    nums[i - 2] = atoi(argv[i]);
  }
  max_prog_1(host, nums, argc - 2);

  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 *max__1_svc(intarr *argp, struct svc_req *rqstp) {

  static int result;

  int a = argp->a, b = argp->b;

  if (a < b)

    result = 1;

  else

    result = 0;

  return (&result);

}
```



```
→  q-05 git:(master) ./max_client
usage: ./max_client <host> [..nums]
→  q-05 git:(master) ./max_client localhost 4 5 6 4 1
Maximum value of array is: 6
→  q-05 git:(master) 
```