**Name: Brijesh Rameshbhai Rohit**

**Admission number: U19CS009**

**DISTRIBUTED SYSTEMS**

**ASSIGNMENT - 06**

**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.**

**CODE :**

| **struct number**  **{**  **int num;**  **};**  **program FACTORIAL\_PROG{**  **version FACTORIAL\_VERS{**  **int factorial(number)=1;**  **}=1;**  **}=0x84562877;** |
| --- |

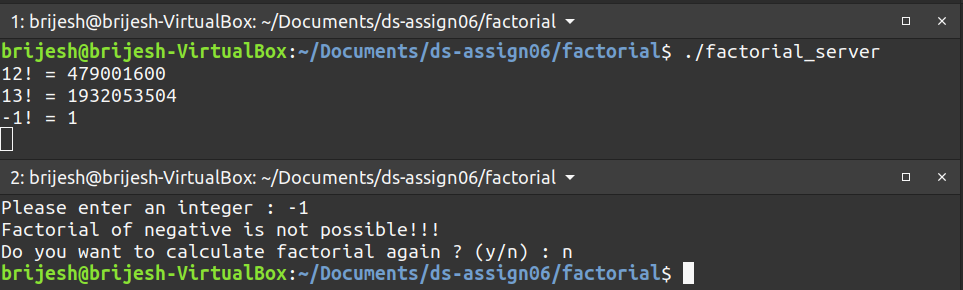
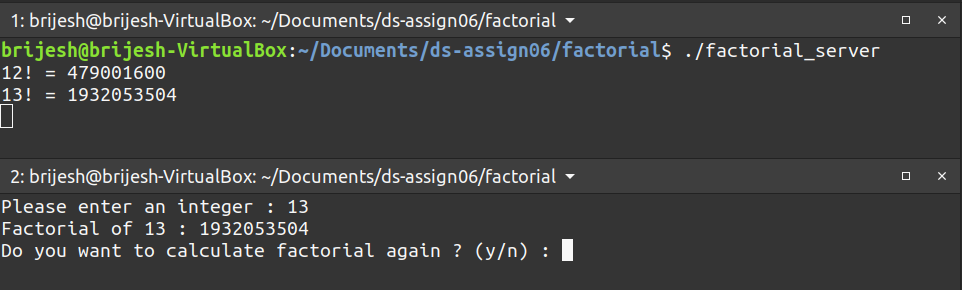
**CODE - SERVER**

| /\*  \* 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(number \*argp, struct svc\_req \*rqstp)  {  static int result;  /\*  \* insert server code here  \*/    int flag = 1;  if(argp->num < 0)  {  printf("Factorial of negative numbers are not calculated!!!\n");  flag = 0;  }  result = 1;  for(int i = 2 ; i <= argp->num ; i++)  result \*= i;  if(flag)  printf("%d! = %d\n", argp->num, result);  return &result;  } |
| --- |

**CODE - CLIENT**

| /\*  \* 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 num1)  {  CLIENT \*clnt;  int \*result\_1;  number factorial\_1\_arg;  #ifndef DEBUG  clnt = clnt\_create (host, FACTORIAL\_PROG, FACTORIAL\_VERS, "udp");  if (clnt == NULL) {  clnt\_pcreateerror (host);  exit (1);  }  #endif /\* DEBUG \*/  int flag = 1;  factorial\_1\_arg.num = num1;  result\_1 = factorial\_1(&factorial\_1\_arg, clnt);  if (result\_1 == (int \*) NULL) {  clnt\_perror (clnt, "call failed");  }  else  {  if(num1 < 0)  {  printf("Factorial of negative is not possible!!!\n");  flag = 0;  }  else  {  printf("Factorial of %d : %d\n", num1, \*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 num1;  char ch = 'y';  while (ch=='y')  {  system("clear");  printf("Please enter an integer : ");  scanf("%d", &num1);  factorial\_prog\_1 (host, num1);  printf("Do you want to calculate factorial again ? (y/n) : ");  scanf("%s", &ch);  }  exit (0);  } |
| --- |

**OUTPUT :**

****

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

**CODE :**

| **struct inputs{**  **float num1;**  **float num2;**  **char operator;**  **};**    **program CALCULATE\_PROG{**  **version CALCULATE\_VER{**  **float CALCULATE(inputs)=1;**  **}=1;**  **}=0x2fffffff;** |
| --- |

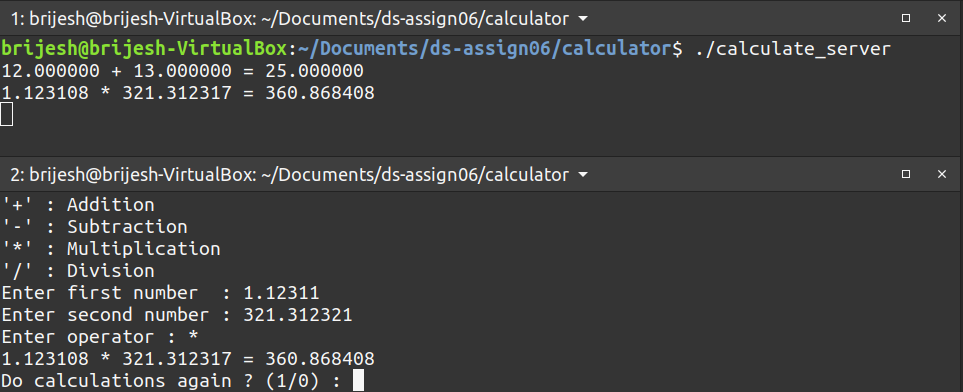
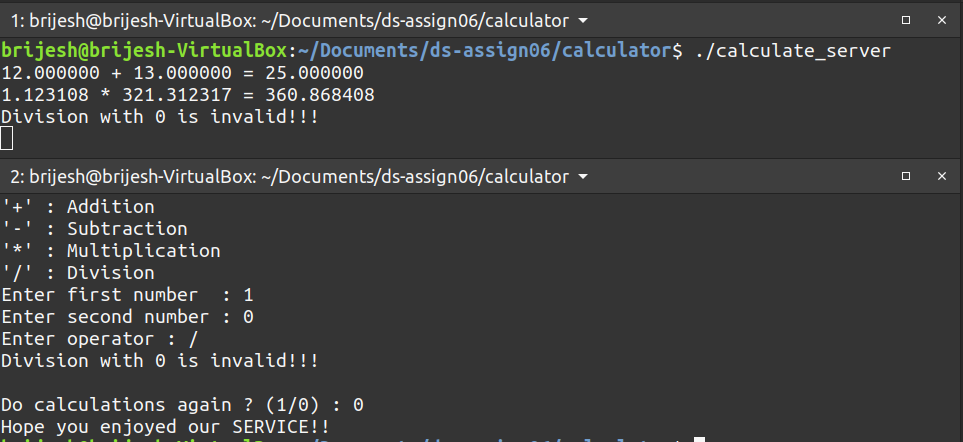
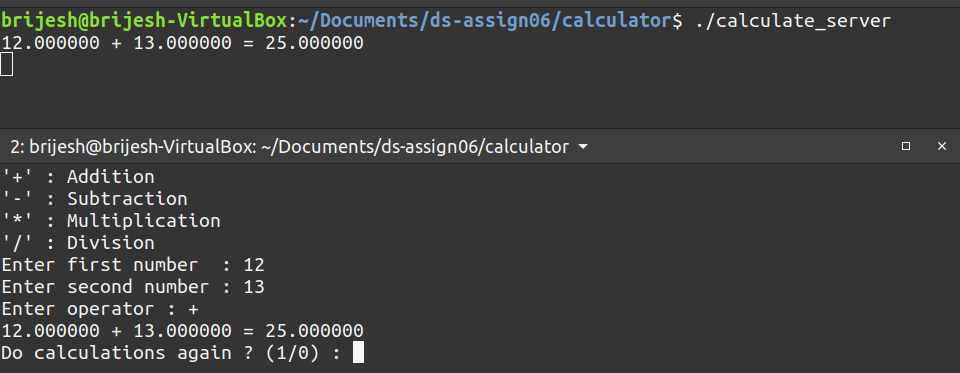
**CODE - SERVER**

| /\*  \* 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 \* calculate\_1\_svc(inputs \*argp, struct svc\_req \*rqstp)  {  static float result;  float a = argp->num1;  float b = argp->num2;  char op = argp->operator;  int flag = 1;  if(op=='+')  result = a+b;  else if(op=='-')  result = a-b;  else if(op=='\*')  result = a\*b;  else  {  if(b==0)  {  printf("Division with 0 is invalid!!!\n");  flag = 0;  }  }    if(flag)  printf("%f %c %f = %f\n",a, op, b, result);    return &result;  } |
| --- |

**CODE - CLIENT**

| /\*  \* 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, float a, float b, char op)  {  CLIENT \*clnt;  float \*result\_1;  inputs calculate\_1\_arg;  #ifndef DEBUG  clnt = clnt\_create (host, CALCULATE\_PROG, CALCULATE\_VER, "udp");  if (clnt == NULL) {  clnt\_pcreateerror (host);  exit (1);  }  #endif /\* DEBUG \*/  calculate\_1\_arg.num1 = a;  calculate\_1\_arg.num2 = b;  calculate\_1\_arg.operator = op;  result\_1 = calculate\_1(&calculate\_1\_arg, clnt);  if (result\_1 == (float \*) NULL) {  clnt\_perror (clnt, "call failed");  }  else  {  int flag = 1;  if(op=='/')  {  if(b==0)  {  printf("Division with 0 is invalid!!!\n");  flag = 0;  }  }  if(flag)  {  printf("%f %c %f = %f",a, op, b, \*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];    char op;  float a, b;  int ch=1;  while(ch)  {  system("clear");  printf("'+' : Addition\n");  printf("'-' : Subtraction\n");  printf("'\*' : Multiplication\n");  printf("'/' : Division\n");  printf("Enter first number : ");  scanf("%f", &a);  printf("Enter second number : ");  scanf("%f", &b);  printf("Enter operator : ");  scanf("%s", &op);  calculate\_prog\_1 (host, a, b, op);  printf("\nDo calculations again ? (1/0) : ");  scanf("%d", &ch);  }  printf("Hope you enjoyed our SERVICE!!\n");  exit (0);  } |
| --- |

**OUTPUT :**

****

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

**CODE :**

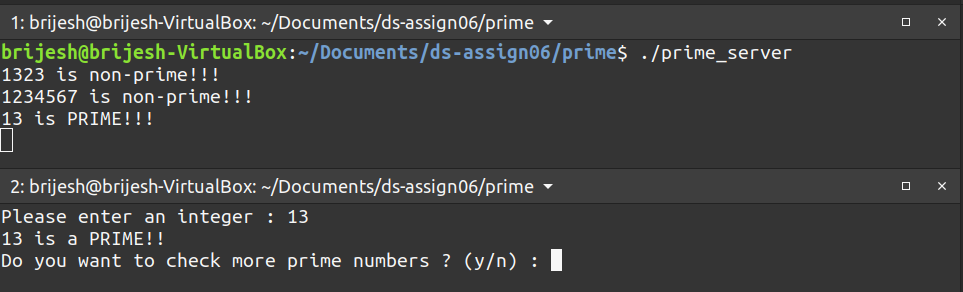
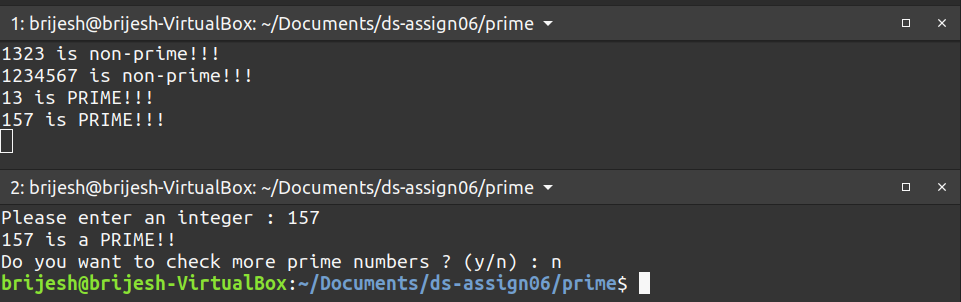
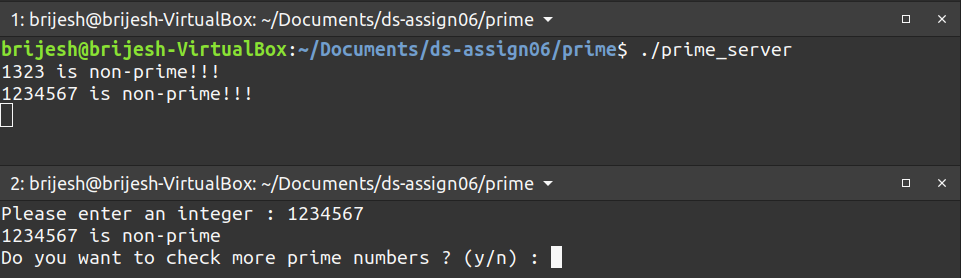
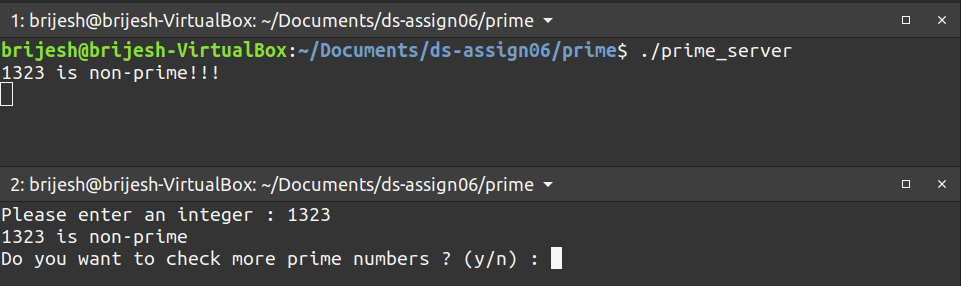
| **struct number{**  **int num;**  **};**  **program PRIME\_PROG{**  **version PRIME\_VERS{**  **int prime(number)=1;**  **}=1;**  **}=0x2f2f2f2f;** |
| --- |

**CODE - SERVER**

| /\*  \* 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(number \*argp, struct svc\_req \*rqstp)  {  static int result;  result=1;  if(argp->num < 1)  {  printf("Entered number is not positive!!\n");  result = 0;  return &result;  }  else if(argp->num == 1)  {  printf("1 is non-prime!!!\n");  result = 0;  return &result;  }  int n = 1;  while(n\*n <= argp->num)  n++;  for(int i = 2 ; i <= n ; i++)  {  if(argp->num%i==0)  {  result = 0;  printf("%d is non-prime!!!\n", argp->num);  return &result;  }  }  printf("%d is PRIME!!!\n", argp->num);  return &result;  } |
| --- |

**CODE - CLIENT**

| /\*  \* 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 num1)  {  CLIENT \*clnt;  int \*result\_1;  number prime\_1\_arg;  #ifndef DEBUG  clnt = clnt\_create (host, PRIME\_PROG, PRIME\_VERS, "udp");  if (clnt == NULL) {  clnt\_pcreateerror (host);  exit (1);  }  #endif /\* DEBUG \*/  int flag = 1;  prime\_1\_arg.num = num1;  result\_1 = prime\_1(&prime\_1\_arg, clnt);  if (result\_1 == (int \*) NULL) {  clnt\_perror (clnt, "call failed");  }  if(num1 < 1)  {  printf("Prime characteristic is only for Positive integers!!!!\n");  }  else  {  if(\*result\_1)  {  printf("%d is a PRIME!!\n", num1);  }  else  {  printf("%d is non-prime\n", num1);  }  }  #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];  char ch='y';  int num;  while(ch=='y')  {  system("clear");  printf("Please enter an integer : ");  scanf("%d", &num);  prime\_prog\_1 (host, num);  printf("Do you want to check more prime numbers ? (y/n) : ");  scanf("%s", &ch);  }    exit (0);  } |
| --- |

**OUTPUT : **

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

**CODE :**

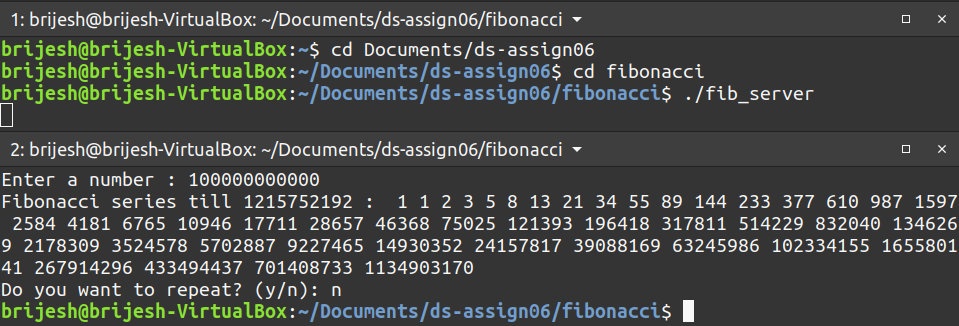
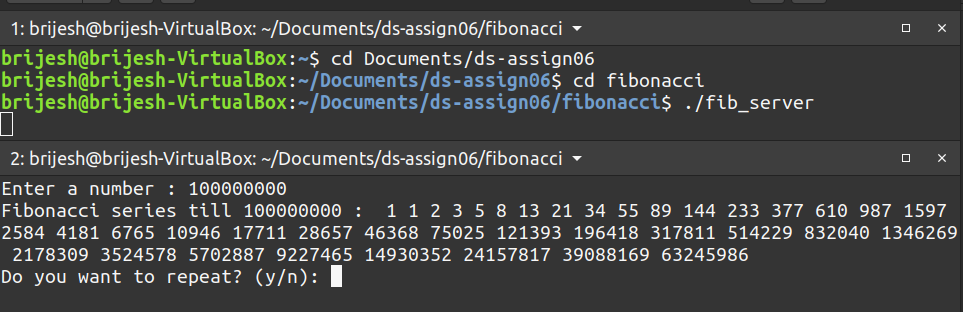
| **struct fibonacci{**  **int n;**  **};**  **program FIB\_PROG{**  **version FIB\_VERS{**  **int fib(fibonacci)=1;**  **}=1;**  **}=0x2f2f2f2f;** |
| --- |

**CODE - SERVER**

| /\*  \* 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"  int \*  fib\_1\_svc(fibonacci \*argp, struct svc\_req \*rqstp)  {  static int result;  int n=argp->n;  if(n<=2)  result=1;  else  {  int cur=1,prev=1,i=2,temp;  while(i<n)  {  temp=cur;  cur=cur+prev;  prev=temp;  i++;  }  result=cur;  }  return &result;  } |
| --- |

**CODE - CLIENT**

| /\*  \* 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;  int \*result\_1;  fibonacci 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.n = 1;  printf("Fibonacci series till %d : ", n);  result\_1 = fib\_1(&fib\_1\_arg, clnt);  if (result\_1 == (int \*) NULL) {  clnt\_perror (clnt, "call failed");  }  else  {  int i = 2;  while (\*result\_1<n)  {  printf(" %d",\*result\_1);  fib\_1\_arg.n=i;  result\_1 = fib\_1(&fib\_1\_arg, clnt);  i++;  }  printf("\n");  }  #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;  char ch='y';  while(ch=='y')  {  system("clear");  printf("Enter a number : ");  scanf("%d",&n);  fib\_prog\_1 (host,n);  printf("Do you want to repeat? (y/n): ");  scanf("%s",&ch);  }  exit (0);  } |
| --- |

**OUTPUT :**

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

**CODE :**

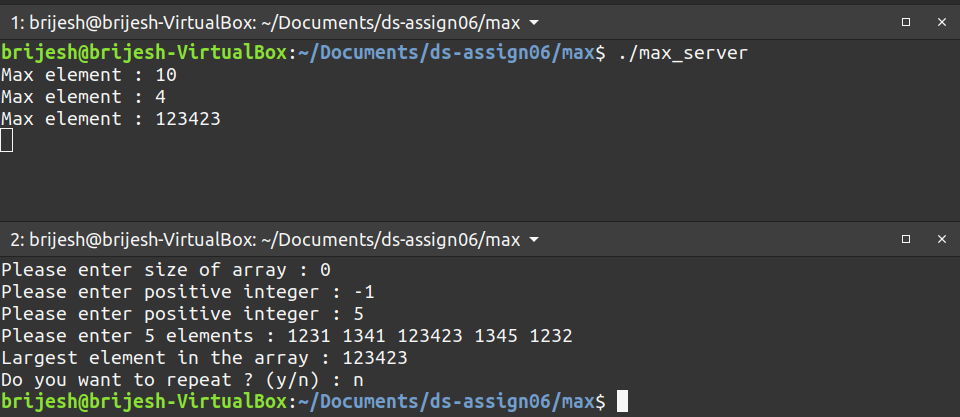
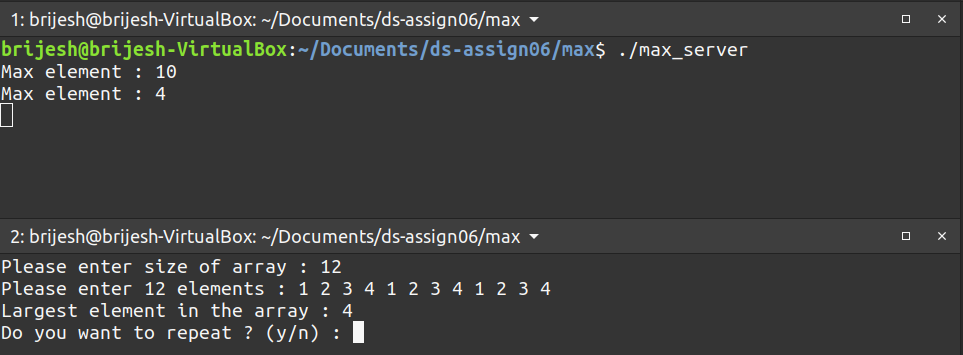
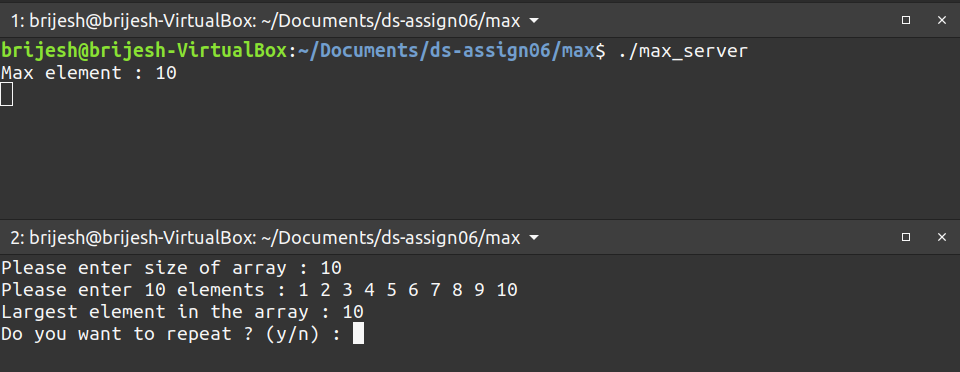
| **struct maxarray{**  **int size;**  **int arr[100];**  **};**  **program MAXARRAY\_PROG{**  **version MAXARRAY\_VERS{**  **int max(maxarray)=1;**  **}=1;**  **}=0x2f2f2f2f;** |
| --- |

**CODE - SERVER**

| /\*  \* 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(maxarray \*argp, struct svc\_req \*rqstp)  {  static int result;  result = argp->arr[0];  int n = argp->size;  for(int i = 1 ; i < n ; i++)  {  if(result < argp->arr[i])  result = argp->arr[i];  }  printf("Max element : %d\n", result);  return &result;  } |
| --- |

**CODE - CLIENT**

| /\*  \* 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  maxarray\_prog\_1(char \*host, int n)  {  CLIENT \*clnt;  int \*result\_1;  maxarray max\_1\_arg;  #ifndef DEBUG  clnt = clnt\_create (host, MAXARRAY\_PROG, MAXARRAY\_VERS, "udp");  if (clnt == NULL) {  clnt\_pcreateerror (host);  exit (1);  }  #endif /\* DEBUG \*/  max\_1\_arg.size = n;  printf("Please enter %d elements : ", n);  for (int i = 0; i < n; i++)  scanf("%d", &max\_1\_arg.arr[i]);  result\_1 = max\_1(&max\_1\_arg, clnt);  if (result\_1 == (int \*) NULL) {  clnt\_perror (clnt, "call failed");  }  else  {  printf("Largest element in the array : %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];  char ch='y';  int n=0;  while (ch=='y')  {  system("clear");  printf("Please enter size of array : ");  scanf("%d", &n);  if(n < 1 || n>100)  {  while (n<1)  {  printf("Please enter positive integer [1-100] : ");  scanf("%d", &n);  }  }  n%=101;  maxarray\_prog\_1 (host, n);  printf("Do you want to repeat ? (y/n) : ");  scanf("%s", &ch);  }  exit (0);  } |
| --- |

**OUTPUT :**