Distributed Systems (CS304)

Assignment - 7

**U19CS012**

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

**List of Programs for RPC**

1.) String is **Palindrome or Not.**

**[q1.x]**

program PALINDROME\_PROG{

    version VERSION1 {

        int palindrome(string s)=1;

    }=1;

}=0x4562877;

**Run Command** : rpcgen –a –C q1.x

* All required files will be created.
* The q1\_client.c and q1\_server.c files would be modified as following:

**[q1\_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* "q1.h"

void

palindrome\_prog\_1(char \*host, char\* str)

{

    CLIENT \*clnt;

    int  \*result\_1;

    char \* palindrome\_1\_arg = str;

*#ifndef* DEBUG

    clnt = clnt\_create (host, PALINDROME\_PROG, VERSION1, "udp");

*if* (clnt == NULL) {

        clnt\_pcreateerror (host);

        exit (1);

    }

*#endif*  */\* DEBUG \*/*

    result\_1 = palindrome\_1(&palindrome\_1\_arg, clnt);

*if* (result\_1 == (int \*) NULL) {

        clnt\_perror (clnt, "call failed");

    }

*if*(\*result\_1 == 1){

        printf("Paildrome\n");

    }

*else*{

        printf("Not Palindrome\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];

    palindrome\_prog\_1 (host,argv[2]);

exit (0);

}

**[q1\_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* "q1.h"

int \*

palindrome\_1\_svc(char \*\*argp, struct svc\_req \*rqstp)

{

    static int  result;

    printf("Palindrome check for %s is called\n",\*argp);

    result = 1;

    int len = strlen(\*argp);

*for*(int i=0;i<len;i++){

*if*((\*argp)[i] != (\*argp)[len-i-1]){

            result = 0;*break*;

        }

    }

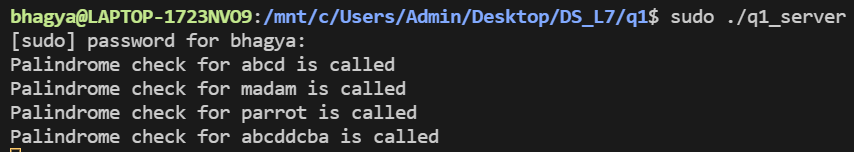
*return* &result;

}

**Run Command** : make –f Makefile.q1

[**Output**]

Server:



Client:



2.) Find out if a given year is a **Lear Year** or not.

**[q2.x]**

program LEAPYEAR\_PROG{

    version VERSION1 {

        int leapyear(int year) = 1;

    } = 1;

} =  0x4562877;

**Run Command** : rpcgen –a –C q2.x

* All required files will be created.
* The q2\_client.c and q2\_server.c files would be modified as following:

**[q2\_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* "q2.h"

void

leapyear\_prog\_1(char \*host,int year)

{

    CLIENT \*clnt;

    int  \*result\_1;

    int  leapyear\_1\_arg = year;

*#ifndef* DEBUG

    clnt = clnt\_create (host, LEAPYEAR\_PROG, VERSION1, "udp");

*if* (clnt == NULL) {

        clnt\_pcreateerror (host);

        exit (1);

    }

*#endif*  */\* DEBUG \*/*

    result\_1 = leapyear\_1(&leapyear\_1\_arg, clnt);

*if* (result\_1 == (int \*) NULL) {

        clnt\_perror (clnt, "call failed");

    }

*if*(\*result\_1 == 1){

        printf("Leap Year\n");

    }*else*{

        printf("Not Leap Year\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];

    int year = atoi(argv[2]);

    leapyear\_prog\_1 (host,year);

exit (0);

}

**[q2\_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* "q2.h"

int \*

leapyear\_1\_svc(int \*argp, struct svc\_req \*rqstp)

{

    static int  result;

    printf("Leap year check for %d is called\n",\*argp);

    result = 0;

*if*(\*argp % 4 == 0){

        result = 1;

    }

*if*(\*argp % 100 == 0){

        result = 0;

    }

*if*(\*argp % 400 == 0){

        result = 1;

    }

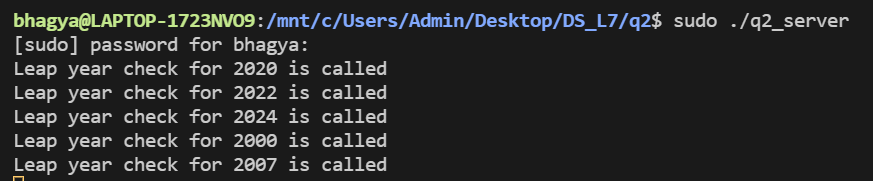
*return* &result;

}

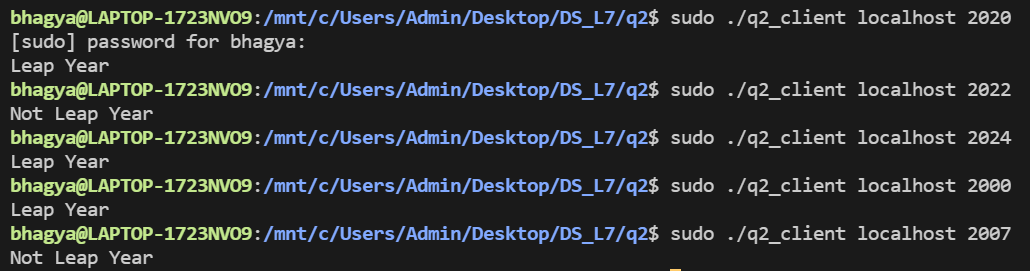
**Run Command** : make –f Makefile.q2

[**Output**]

Server:



Client:



3.) Find out the **GCD** of a given number.

**[q3.x]**

struct intpair{

    int a;

    int b;

};

program GCD\_PROG{

    version VERSION1 {

        int gcd(intpair p) = 1;

    } = 1;

} =  0x4562877;

**Run Command** : rpcgen –a –C q3.x

* All required files will be created.
* The q3\_client.c and q3\_server.c files would be modified as following:

**[q3\_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* "q3.h"

void

gcd\_prog\_1(char \*host,int a,int b)

{

    CLIENT \*clnt;

    int  \*result\_1;

    intpair  gcd\_1\_arg = {a,b};

*#ifndef* DEBUG

    clnt = clnt\_create (host, GCD\_PROG, VERSION1, "udp");

*if* (clnt == NULL) {

        clnt\_pcreateerror (host);

        exit (1);

    }

*#endif*  */\* DEBUG \*/*

    result\_1 = gcd\_1(&gcd\_1\_arg, clnt);

*if* (result\_1 == (int \*) NULL) {

        clnt\_perror (clnt, "call failed");

    }

    printf("gcd : %d\n",\*result\_1);

*#ifndef* DEBUG

    clnt\_destroy (clnt);

*#endif*   */\* DEBUG \*/*

}

int

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

{

    char \*host;

*if* (argc < 4) {

        printf ("usage: %s server\_host\n", argv[0]);

        exit (1);

    }

    host = argv[1];

    int a = atoi(argv[2]);

    int b = atoi(argv[3]);

    gcd\_prog\_1 (host,a,b);

exit (0);

}

**[q3\_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* "q3.h"

int gcd\_s(int a,int b){

*if*(a==0){

*return* b;

    }

*if*(a>b){

        gcd\_s(a%b,b);

    }

*else*{

        gcd\_s(b%a,a);

    }

}

int \*

gcd\_1\_svc(intpair \*argp, struct svc\_req \*rqstp)

{

    static int  result;

    printf("GCD for %d and %d is called\n",argp->a,argp->b);

    result = gcd\_s(argp->a,argp->b);

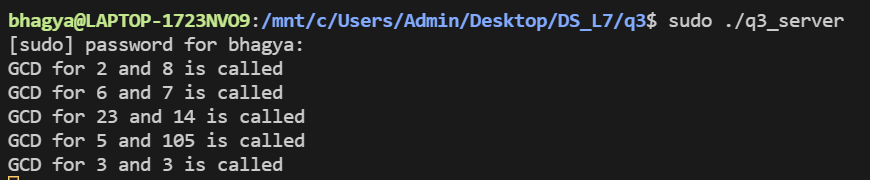
*return* &result;

}

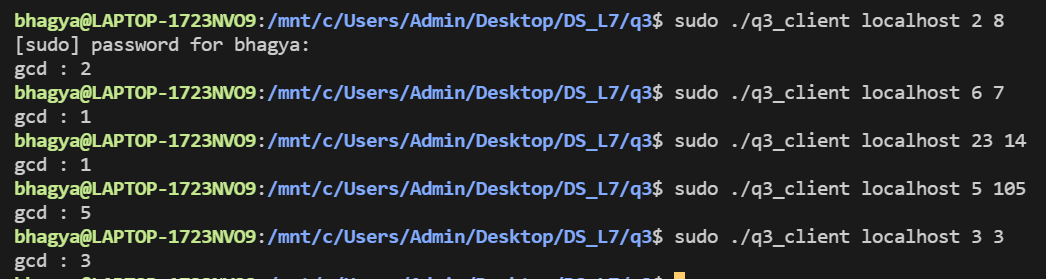
**Run Command** : make –f Makefile.q3

[**Output**]

Server:



Client:



4.) Find out the **Square root** of a given number.

**[q4.x]**

program SQRT\_PROG{

    version VERSION1 {

        float sqrt(float n) = 1;

    } = 1;

} =  0x4562877;

**Run Command** : rpcgen –a –C q4.x

* All required files will be created.
* The q4\_client.c and q4\_server.c files would be modified as following:

**[q4\_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* "q4.h"

void

sqrt\_prog\_1(char \*host,float n)

{

    CLIENT \*clnt;

    float  \*result\_1;

    float  sqrt\_1\_arg=n;

*#ifndef* DEBUG

    clnt = clnt\_create (host, SQRT\_PROG, VERSION1, "udp");

*if* (clnt == NULL) {

        clnt\_pcreateerror (host);

        exit (1);

    }

*#endif*  */\* DEBUG \*/*

    result\_1 = sqrt\_1(&sqrt\_1\_arg, clnt);

*if* (result\_1 == (float \*) NULL) {

        clnt\_perror (clnt, "call failed");

    }

    printf("sqrt : %f\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\n", argv[0]);

        exit (1);

    }

    host = argv[1];

    float n = atof(argv[2]);

    sqrt\_prog\_1 (host,n);

exit (0);

}

**[q4\_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* "q4.h"

float \*

sqrt\_1\_svc(float \*argp, struct svc\_req \*rqstp)

{

    static float  result;

*/\**

*\* insert server code here*

*\*/*

    printf("Sqrt for %f is called\n",\*argp);

    float err = 0.00001f;

    float a = \*argp;

    float b = 2.0f;

*if*(abs(b-a)<=err) b = 1.5f;

*while*((b-a)>err || (a-b)>err){

        a = \*argp/b;

        b = (a+b)/2;

    }

    result = a;

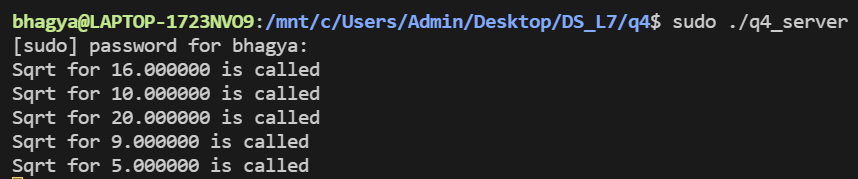
*return* &result;

}

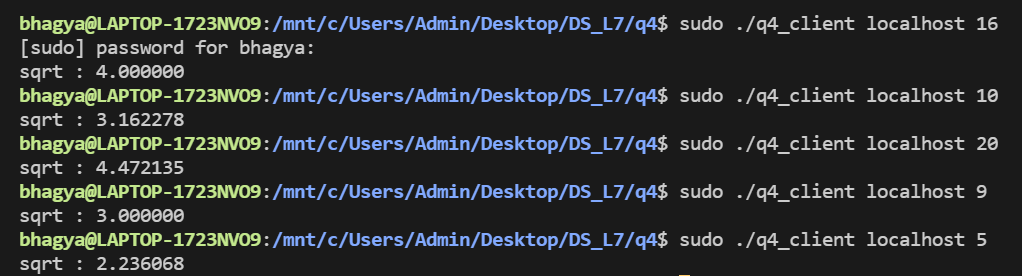
**Run Command** : make –f Makefile.q4

[**Output**]

Server:



Client:



5.) **Swap two variables** without using the 3rd variable.

**[q5.x]**

struct intpair{

    int a;

    int b;

};

program SWAP\_PROG{

    version VERSION1 {

        intpair swap(intpair p) = 1;

    } = 1;

} =  0x4562877;

**Run Command** : rpcgen –a –C q5.x

* All required files will be created.
* The swap\_client.c and swap\_server.c files would be modified as following:

**[q5\_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* "q5.h"

void

swap\_prog\_1(char \*host,int a,int b)

{

    CLIENT \*clnt;

    intpair  \*result\_1;

    intpair  swap\_1\_arg = {a,b};

*#ifndef* DEBUG

    clnt = clnt\_create (host, SWAP\_PROG, VERSION1, "udp");

*if* (clnt == NULL) {

        clnt\_pcreateerror (host);

        exit (1);

    }

*#endif*  */\* DEBUG \*/*

    result\_1 = swap\_1(&swap\_1\_arg, clnt);

*if* (result\_1 == (intpair \*) NULL) {

        clnt\_perror (clnt, "call failed");

    }

    printf("a: %d , b: %d \n",result\_1->a,result\_1->b);

*#ifndef* DEBUG

    clnt\_destroy (clnt);

*#endif*   */\* DEBUG \*/*

}

int

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

{

    char \*host;

*if* (argc < 4) {

        printf ("usage: %s server\_host\n", argv[0]);

        exit (1);

    }

    host = argv[1];

    int a = atoi(argv[2]);

    int b = atoi(argv[3]);

    swap\_prog\_1 (host,a,b);

exit (0);

}

**[q5\_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* "q5.h"

intpair \*

swap\_1\_svc(intpair \*argp, struct svc\_req \*rqstp)

{

    static intpair  result;

*/\**

*\* insert server code here*

*\*/*

    printf("Swap called\n");

    result.a = argp->b;

    result.b = argp->a;

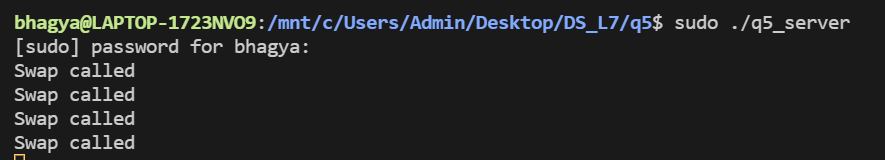
*return* &result;

}

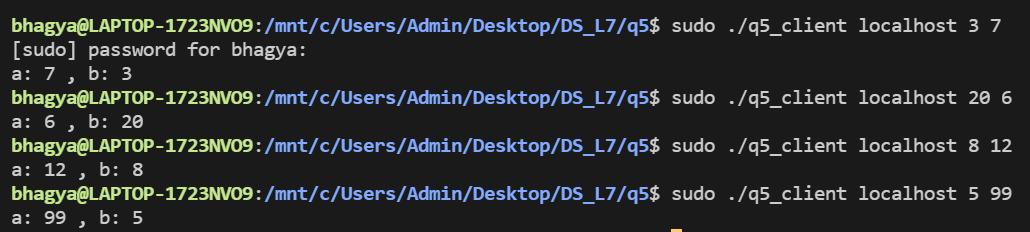
**Run Command** : make –f Makefile.q5

[**Output**]

Server:



Client:



6.) Calculate **Maximum, Minimum, average** of given array.

**[q6.x]**

struct arraysize{

    int\* arr;

    int size;

};

program ARR\_PROG{

    version VERSION1 {

        arraysize arrop(arraysize arr) = 1;

    } = 2;

} =  0x20000005;

**Run Command** : rpcgen –a –C q6.x

* All required files will be created.
* The q6\_client.c and q6\_server.c files would be modified as following:

**[q6\_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* "q6.h"

void

arr\_prog\_2(char \*host,int n,int\* arr)

{

    CLIENT \*clnt;

    arraysize  \*result\_1;

    arraysize  arrop\_2\_arg = {arr,n};

*#ifndef* DEBUG

    clnt = clnt\_create (host, ARR\_PROG, VERSION1, "udp");

*if* (clnt == NULL) {

        clnt\_pcreateerror (host);

        exit (1);

    }

*#endif*  */\* DEBUG \*/*

    result\_1 = arrop\_2(&arrop\_2\_arg, clnt);

*if* (result\_1 == (arraysize \*) NULL) {

        clnt\_perror (clnt, "call failed");

    }

    printf("Min:%d\nMax:%d\nAvg:%d\n",result\_1->arr[0],result\_1->arr[1],result\_1->arr[2]);

*#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 = atoi(argv[2]);

    int arr[n];

*for*(int i=0;i<n;i++)arr[i] = atoi(argv[i+3]);

    arr\_prog\_2 (host,n,arr);

exit (0);

}

**[q6\_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* "q6.h"

arraysize \*

arrop\_2\_svc(arraysize \*argp, struct svc\_req \*rqstp)

{

    static arraysize  result;

    printf("Maximum,Minimum,Average called for an array\n");

    result.size = 3;

    static int arr[3];

    arr[0] = INT\_MAX;

    arr[1] = INT\_MIN;

    arr[2] = 0;

*for*(int i=0;i<argp->size;i++){

*if*(arr[0] > argp->arr[i]){

            arr[0] = argp->arr[i];

        }

*if*(arr[1] < argp->arr[i]){

            arr[1] = argp->arr[i];

        }

        arr[2] += argp->arr[i];

    }

    arr[2] /= argp->size;

    result.arr = arr;

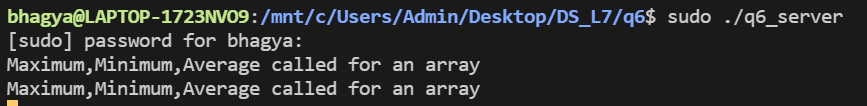
*return* &result;

}

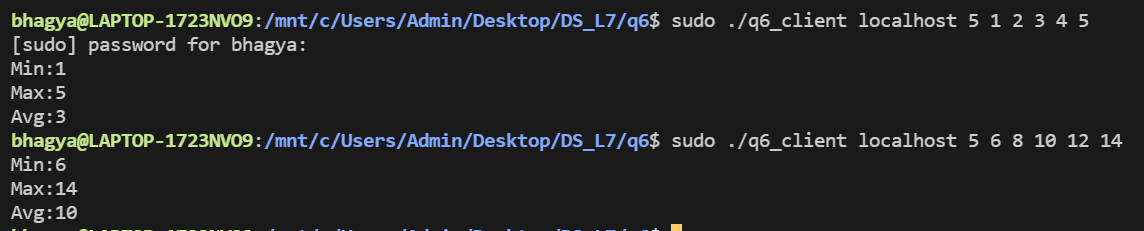
**Run Command** : make –f Makefile.q6

[**Output**]

Server:



Client:



7.) Compare the given two strings.

**[q7.x]**

struct strpair{

    int len1;

    char\* a;

    int len2;

    char\* b;

};

program ARR\_PROG{

    version VERSION1 {

        int cmp(strpair p) = 1;

    } = 2;

} =  0x20000007;

**Run Command** : rpcgen –a –C q7.x

* All required files will be created.
* The compare\_client.c and compare\_server.c files would be modified as following:

**[q7\_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* "q7.h"

void

arr\_prog\_2(char \*host,char\* a,char\* b)

{

    CLIENT \*clnt;

    int  \*result\_1;

    strpair  cmp\_2\_arg={strlen(a)+1,a,strlen(b)+1,b};

*#ifndef* DEBUG

    clnt = clnt\_create (host, ARR\_PROG, VERSION1, "udp");

*if* (clnt == NULL) {

        clnt\_pcreateerror (host);

        exit (1);

    }

*#endif*  */\* DEBUG \*/*

    result\_1 = cmp\_2(&cmp\_2\_arg, clnt);

*if* (result\_1 == (int \*) NULL) {

        clnt\_perror (clnt, "call failed");

    }

    printf("%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];

    arr\_prog\_2 (host,argv[2],argv[3]);

exit (0);

}

**[q7\_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* "q7.h"

int \*

cmp\_2\_svc(strpair \*argp, struct svc\_req \*rqstp)

{

    static int  result;

    printf("Compare called for two strings\n");

    result = strcmp(argp->a,argp->b);

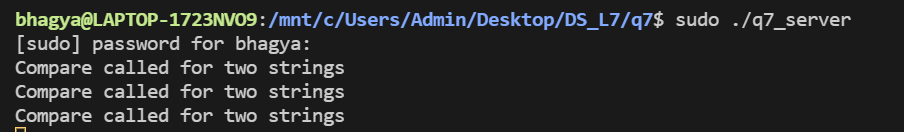
*return* &result;

}

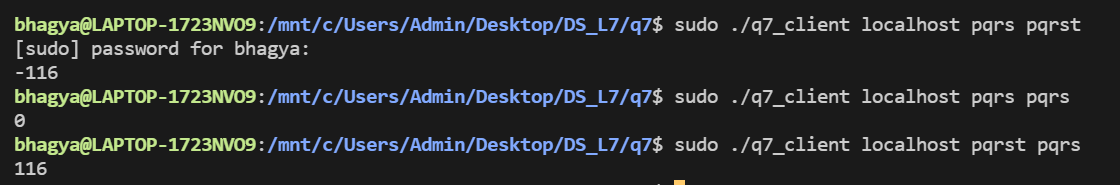
**Run Command** : make –f Makefile.q7

[**Output**]

Server:



Client:



8.) Find out whether a given string is **substring or not**.

**[q8.x]**

struct strpair{

    int len1;

    char\* a;

    int len2;

    char\* b;

};

program ARR\_PROG{

    version VERSION1 {

        int substring(strpair p) = 1;

    } = 2;

} =  0x20000007;

**Run Command** : rpcgen –a –C q8.x

* All required files will be created.
* The q8\_client.c and q8\_server.c files would be modified as following:

**[q8\_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* "q8.h"

void

arr\_prog\_2(char \*host,char\* a,char\* b)

{

    CLIENT \*clnt;

    int  \*result\_1;

    strpair  substring\_2\_arg = {strlen(a)+1,a,strlen(b)+1,b};

*#ifndef* DEBUG

    clnt = clnt\_create (host, ARR\_PROG, VERSION1, "udp");

*if* (clnt == NULL) {

        clnt\_pcreateerror (host);

        exit (1);

    }

*#endif*  */\* DEBUG \*/*

    result\_1 = substring\_2(&substring\_2\_arg, clnt);

*if* (result\_1 == (int \*) NULL) {

        clnt\_perror (clnt, "call failed");

    }

*if*(\*result\_1 == 1){

        printf("Is a substring\n");

    }

*else* printf("Not a substring\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];

    arr\_prog\_2 (host,argv[2],argv[3]);

exit (0);

}

**[q8\_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* "q8.h"

int \*

substring\_2\_svc(strpair \*argp, struct svc\_req \*rqstp)

{

    static int  result;

    printf("Substring check is called for two strinsgs\n");

    int l1 = argp->len1 -1;

    int l2 = argp->len2 -1;

    result = 0;

*if*(l1 < l2){

*return* &result;

    }

    char temp[l2+1];

    temp[l2]=0;

*for*(int i=0;i<=l1-l2;i++){

        memcpy(temp,&argp->a[i],l2);

        result |= !strcmp(temp,argp->b);

    }

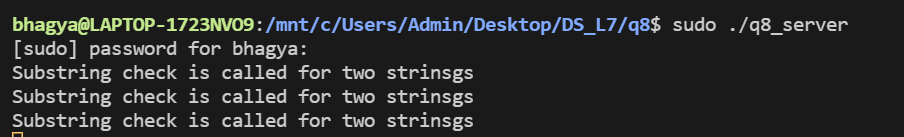
*return* &result;

}

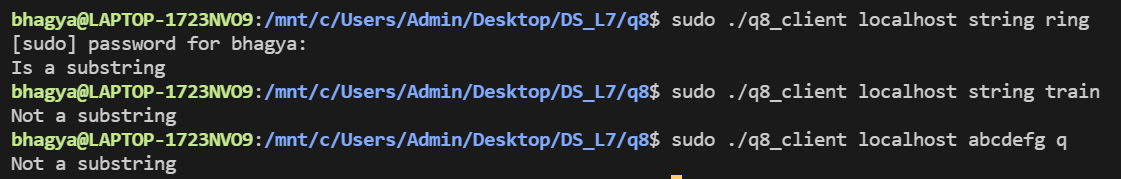
**Run Command** : make –f Makefile.q8

[**Output**]

Server:



Client:



9.) **Concatenate** the two strings.

**[q9.x]**

struct strpair{

    int len1;

    char\* a;

    int len2;

    char\* b;

};

program CONCAT\_PROG{

    version VERSION1 {

        string concatenate(strpair p) = 1;

    } = 1;

} =  0x4562877;

**Run Command** : rpcgen –a –C q9.x

* All required files will be created.
* The q9\_client.c and q9\_server.c files would be modified as following:

**[q9\_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* "q9.h"

void

concat\_prog\_1(char \*host,char\* a,char\* b)

{

    CLIENT \*clnt;

    char \* \*result\_1;

    strpair  concatenate\_1\_arg = {strlen(a)+1,a,strlen(b)+1,b};

*#ifndef* DEBUG

    clnt = clnt\_create (host, CONCAT\_PROG, VERSION1, "udp");

*if* (clnt == NULL) {

        clnt\_pcreateerror (host);

        exit (1);

    }

*#endif*  */\* DEBUG \*/*

    result\_1 = concatenate\_1(&concatenate\_1\_arg, clnt);

*if* (result\_1 == (char \*\*) NULL) {

        clnt\_perror (clnt, "call failed");

    }

    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];

    concat\_prog\_1 (host,argv[2],argv[3]);

exit (0);

}

**[q9\_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* "q9.h"

char \*\*

concatenate\_1\_svc(strpair \*argp, struct svc\_req \*rqstp)

{

    static char \* result;

    printf("Concatenation of two strings is called. \n");

    result = malloc(argp->len1 + argp->len2 +1);

    memcpy(result,argp->a,argp->len1);

    strcat(result,argp->b);

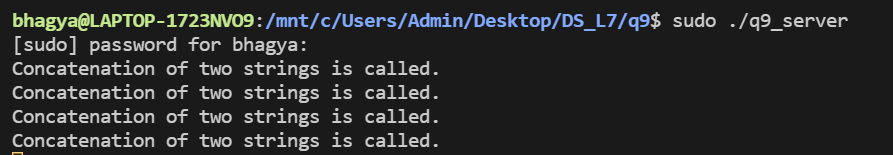
*return* &result;

}

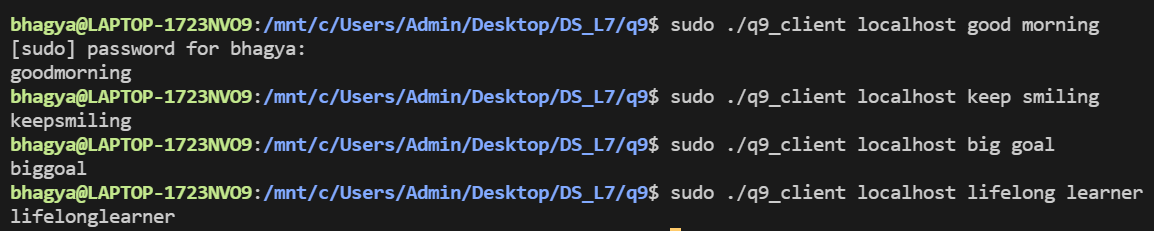
**Run Command** : make –f Makefile.q9

[**Output**]

Server:



Client:



10.) **Reverse** the elements of an array.

**[q10.x]**

struct arraysize{

    int\* arr;

    int size;

};

program ARR\_PROG{

    version VERSION1 {

        arraysize reverse(arraysize arr) = 1;

    } = 2;

} =  0x20000005;

**Run Command** : rpcgen –a –C q10.x

* All required files will be created.
* The q10\_client.c and q10\_server.c files would be modified as following:

**[q10\_client.c]**

*#include* "q10.h"

void

arr\_prog\_2(char \*host,int n,int\* arr)

{

    CLIENT \*clnt;

    arraysize  \*result\_1;

    arraysize  reverse\_2\_arg = {arr,n};

*#ifndef* DEBUG

    clnt = clnt\_create (host, ARR\_PROG, VERSION1, "udp");

*if* (clnt == NULL) {

        clnt\_pcreateerror (host);

        exit (1);

    }

*#endif*  */\* DEBUG \*/*

    result\_1 = reverse\_2(&reverse\_2\_arg, clnt);

*if* (result\_1 == (arraysize \*) NULL) {

        clnt\_perror (clnt, "call failed");

    }

*for*(int i=0;i<result\_1->size;i++)printf("%d ",result\_1->arr[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 = atoi(argv[2]);

    int arr[n];

*for*(int i=0;i<n;i++)arr[i] = atoi(argv[i+3]);

    arr\_prog\_2 (host,n,arr);

exit (0);

}

**[q10\_server.c]**

*#include* "q10.h"

arraysize \*

reverse\_2\_svc(arraysize \*argp, struct svc\_req \*rqstp)

{

    static arraysize  result;

    printf("Reversing the array...\n");

    result.size = argp->size;

    int\* arr = malloc(argp->size \* sizeof(int));

*for*(int i=0;i<result.size;i++)arr[i] = argp->arr[result.size-i-1];

    result.arr = arr;

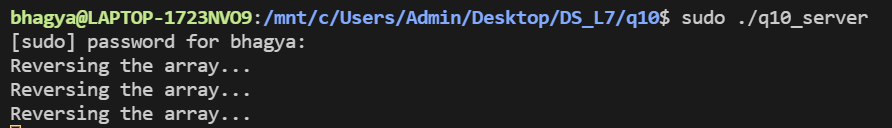
*return* &result;

}

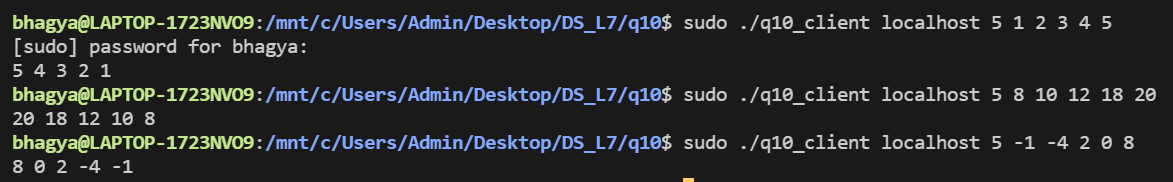
**Run Command** : make –f Makefile.q10

[**Output**]

Server:



Client:



**SUBMITTED BY**: U19CS012

BHAGYA VINOD RANA