

DS Assignment 6

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. String is palindrome or not.

Source Code:

palindrome.x

```
struct intpair {
    char ch[100];
};

program PALINDROME_PROG
{
    version PALINDROME_VERS
    {
        int PALINDROME(intpair) = 1;
    } = 1;
} = 0x23451111;
```

palindrome_client.c

```
#include "palindrome.h"
#include<string.h>

void palindrome_prog_1(char *host, char str[])
{
    CLIENT *clnt;
    int *result_1;
    intpair palindrome_1_arg;

#ifdef DEBUG
    clnt = clnt_create (host, PALINDROME_PROG, PALINDROME_VERS, "udp");
    if (clnt == NULL) {
        clnt_pcreateerror (host);
        exit (1);
    }
#endif /* DEBUG */
    strcpy(palindrome_1_arg.ch, str);
    result_1 = palindrome_1(&palindrome_1_arg, clnt);
    if (result_1 == (int *) NULL) {
        clnt_perror (clnt, "call failed");
    }
    else{
```

```

        if(*result_1==0){
printf("It is a palindrome!\n");
        }
        else{
printf("It is not a palindrome!\n");
        }
    }
#ifdef DEBUG
    clnt_destroy (clnt);
#endif /* DEBUG */
}

int main (int argc, char *argv[])
{
    char *host;
    char ch[100];
    if (argc < 2) {
        printf ("usage: %s server_host\n", argv[0]);
        exit (1);
    }
    printf("Enter the String: ");
    gets(ch);
    host = argv[1];
    palindrome_prog_1 (host,ch);
exit (0);
}

```

palindrome_server.c

```

#include "palindrome.h"
#include<string.h>

int * palindrome_1_svc(intpair *argp, struct svc_req *rqstp)
{
    static int result;
    int i,length,flag=0;
    length= strlen(argp->ch);
    printf("The received string is ");
    puts(argp->ch);
    for(i=0; i<length/2; i++){
        if(argp->ch[i] != argp->ch[length-i-1]){
            flag=1;
            break;
        }
    }
    result=flag;
    return &result;
}

```

Output:

Client Side

```
himani@Himani: ~/Desktop/DS/palindrome
himani@Himani:~/Desktop/DS/palindrome$ ./palindrome_client localhost 12 13
Enter the String: abccba
It is a palindrome!
himani@Himani:~/Desktop/DS/palindrome$ ./palindrome_client localhost 12 13
Enter the String: congratulations
It is not a palindrome!
himani@Himani:~/Desktop/DS/palindrome$ ./palindrome_client localhost 12 13
Enter the String: yuiiuy
It is a palindrome!
himani@Himani:~/Desktop/DS/palindrome$
```

Server Side

```
himani@Himani: ~/Desktop/DS/palindrome
himani@Himani:~/Desktop/DS/palindrome$ make -f Makefile.palindrome
cc -g -c -o palindrome_clnt.o palindrome_clnt.c
cc -g -c -o palindrome_client.o palindrome_client.c
palindrome_client.c: In function 'main':
palindrome_client.c:45:2: warning: implicit declaration of function 'gets'; did
you mean 'fgets'? [-Wimplicit-function-declaration]
   45 |     gets(ch);
      |     ^~~~~
      |     fgets
cc -g -c -o palindrome_xdr.o palindrome_xdr.c
cc -g -o palindrome_client palindrome_clnt.o palindrome_client.o palindrome_xdr.o -lnsl
/usr/bin/ld: palindrome_client.o: in function 'main':
/home/himani/Desktop/DS/palindrome/palindrome_client.c:45: warning: the 'gets' f
unction is dangerous and should not be used.
cc -g -o palindrome_server palindrome_svc.o palindrome_server.o palindrome_xdr.o -lnsl
himani@Himani:~/Desktop/DS/palindrome$ ./palindrome_server
The received string is abccba
The received string is congratulations
The received string is yuiiuy
```

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

Source Code:

leap.x

```
struct intpair {
    int a;
};

program LEAP_PROG
{
    version LEAP_VERS
    {
        int LEAP(intpair) = 1;
    } = 1;
```

```
} = 0x23451111;
```

leap_client.c

```
#include "leap.h"

void leap_prog_1(char *host, int y)
{
    CLIENT *clnt;
    int *result_1;
    intpair leap_1_arg;

#ifdef DEBUG
    clnt = clnt_create (host, LEAP_PROG, LEAP_VERS, "udp");
    if (clnt == NULL) {
        clnt_pcreateerror (host);
        exit (1);
    }
#endif /* DEBUG */
    leap_1_arg.a=y;
    result_1 = leap_1(&leap_1_arg, clnt);
    if (result_1 == (int *) NULL) {
        clnt_perror (clnt, "call failed");
    }
    else{
        if(*result_1==1){
            printf("It is a leap year!\n");
        }
        else{
            printf("It is not a leap year!\n");
        }
    }

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

int main (int argc, char *argv[])
{
    char *host;
    int y;
    if (argc < 2) {
        printf ("usage: %s server_host\n", argv[0]);
        exit (1);
    }
    host = argv[1];
    printf("Enter a year: ");
    scanf("%d", &y);
    leap_prog_1 (host,y);
    exit (0);
}
```

leap_server.c

```
#include "leap.h"

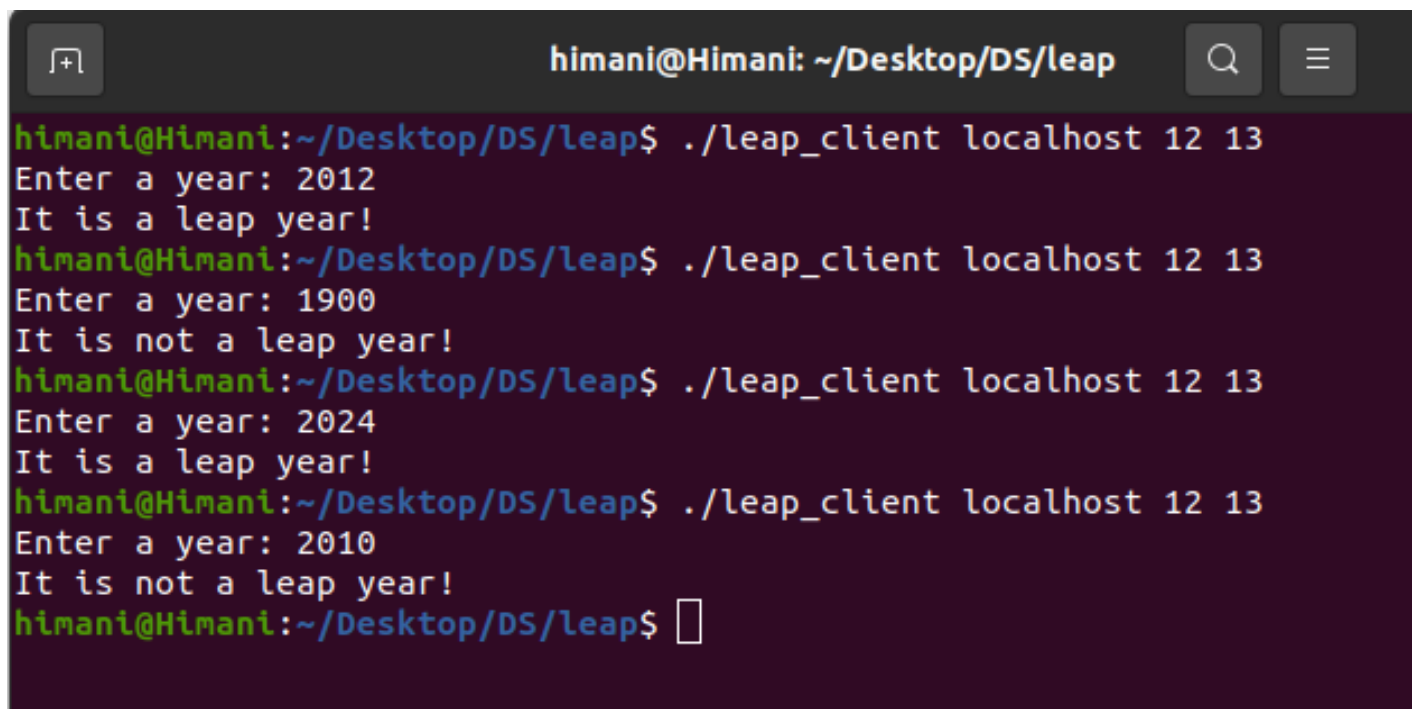
int * leap_1_svc(intpair *argp, struct svc_req *rqstp)
{
    static int result;
    int flag = 0;
    int year= argp->a;
    int ans = 0;
    printf("Server Received the request\n");

    if (year % 100 == 0){
        if (year % 400 == 0){
            ans = 1;
        }
    }else if(year % 4 == 0){
        ans = 1;
    }

    result = ans;
    return &result;
}
```

Output:

Client Side



A terminal window titled 'himani@Himani: ~/Desktop/DS/leap' showing the execution of the leap_client program. The user enters the command './leap_client localhost 12 13' and provides input years. The program outputs whether each year is a leap year.

```
himani@Himani:~/Desktop/DS/leap$ ./leap_client localhost 12 13
Enter a year: 2012
It is a leap year!
himani@Himani:~/Desktop/DS/leap$ ./leap_client localhost 12 13
Enter a year: 1900
It is not a leap year!
himani@Himani:~/Desktop/DS/leap$ ./leap_client localhost 12 13
Enter a year: 2024
It is a leap year!
himani@Himani:~/Desktop/DS/leap$ ./leap_client localhost 12 13
Enter a year: 2010
It is not a leap year!
himani@Himani:~/Desktop/DS/leap$
```

Server Side

```
himani@Himani: ~/Desktop/DS/leap
himani@Himani:~/Desktop/DS/leap$ make -f Makefile.leap
cc -g      -c -o leap_xdr.o leap_xdr.c
cc -g      -o leap_client leap_clnt.o leap_client.o leap_xdr.o -lnsl
cc -g      -c -o leap_svc.o leap_svc.c
cc -g      -c -o leap_server.o leap_server.c
cc -g      -o leap_server leap_svc.o leap_server.o leap_xdr.o -lnsl
himani@Himani:~/Desktop/DS/leap$ ./leap_server
Server Received the request
Server Received the request
Server Received the request
Server Received the request
█
```

3. Find out the GCD of a given number.

Source Code:

gcd.x

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

program GCD_PROG
{
    version GCD_VERS
    {
        int GCD(intpair) = 1;
    } = 1;
} = 0x23451111;
```

gcd_client.c

```
#include "gcd.h"

void gcd_prog_1(char *host,int num1,int num2)
{
    CLIENT *clnt;
    int *result_1;
    intpair gcd_1_arg;

#ifdef DEBUG
    clnt = clnt_create (host, GCD_PROG, GCD_VERS, "udp");
    if (clnt == NULL) {
        clnt_pcreateerror (host);
        exit (1);
    }
#endif /* DEBUG */

    gcd_1_arg.a = num1;
```



```
himani@Himani: ~/Desktop/DS/gcd
himani@Himani:~/Desktop/DS/gcd$ ./gcd_client localhost 12 13
Enter the two numbers : 4 5
The GCD is 1
himani@Himani:~/Desktop/DS/gcd$ ./gcd_client localhost 12 13
Enter the two numbers : 100 10
The GCD is 10
himani@Himani:~/Desktop/DS/gcd$ ./gcd_client localhost 12 13
Enter the two numbers : 12 8
The GCD is 4
himani@Himani:~/Desktop/DS/gcd$
```

Server Side

```
himani@Himani: ~/Desktop/DS/gcd
himani@Himani:~/Desktop/DS/gcd$ make -f Makefile.gcd
cc -g -c -o gcd_clnt.o gcd_clnt.c
cc -g -c -o gcd_client.o gcd_client.c
cc -g -c -o gcd_xdr.o gcd_xdr.c
cc -g -o gcd_client gcd_clnt.o gcd_client.o gcd_xdr.o -lnsl
cc -g -c -o gcd_svc.o gcd_svc.c
cc -g -c -o gcd_server.o gcd_server.c
cc -g -o gcd_server gcd_svc.o gcd_server.o gcd_xdr.o -lnsl
himani@Himani:~/Desktop/DS/gcd$ ./gcd_server

The two numbers sent from client are 4 and 5

The two numbers sent from client are 100 and 10

The two numbers sent from client are 12 and 8

```

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

Source Code:

sq.x

```
struct intpair {
    double a;
};

program SQUARE_PROG
{
    version SQUARE_VERS
    {
        double SQUARE(intpair) = 1;
    } = 1;
}
```



```
} = 0x23451111;
```

sq_client.c

```
#include "sq.h"

void square_prog_1(char *host, double num)
{
    CLIENT *clnt;
    double *result_1;
    intpair square_1_arg;

#ifdef DEBUG
    clnt = clnt_create (host, SQUARE_PROG, SQUARE_VERS, "udp");
    if (clnt == NULL) {
        clnt_pcreateerror (host);
        exit (1);
    }
#endif /* DEBUG */
    square_1_arg.a = num;
    result_1 = square_1(&square_1_arg, clnt);
    if (result_1 == (double *) NULL) {
        clnt_perror (clnt, "call failed");
    } else {
        printf("The square root is : %lf\n", *result_1);
    }
#ifdef DEBUG
    clnt_destroy (clnt);
#endif /* DEBUG */
}

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

    if (argc < 2) {
        printf ("usage: %s server_host\n", argv[0]);
        exit (1);
    }
    host = argv[1];
    printf("Enter the number whose square root is to be found: ");
    scanf("%lf",&num);
    square_prog_1 (host,num);
    exit (0);
}
```

sq_server.c

```
#include "sq.h"
#include <math.h>
```

```
double *
square_1_svc(intpair *argp, struct svc_req *rqstp)
{
    static double result;
    double num = argp->a;

    printf("\nNumber received is %lf\n",num);
    double ans = sqrt(num);
    result = ans;
    return &result;
}
```

Since we are using the “math.h” header file, we have to modify the makefile as the mathematical functions (as declared in math.h) are bundled separately in the mathematical library libm. Therefore, if any of those functions are used, the linker must be given the directive -lm. We have to modify:

LDLIBS += -lnsl

Which will be changed to:

LDLIBS += -lnsl -lm

Output:

Client Side

A terminal window titled 'himani@Himani: ~/Desktop/DS/square' showing the execution of a client program. The user runs './sq_client localhost 12 13' four times, entering different numbers (16, 2, 18, 100) and receiving their square roots (4.000000, 1.414214, 4.242641, 10.000000) as output.

```
himani@Himani:~/Desktop/DS/square$ ./sq_client localhost 12 13
Enter the number whose square root is to be found: 16
The square root is : 4.000000
himani@Himani:~/Desktop/DS/square$ ./sq_client localhost 12 13
Enter the number whose square root is to be found: 2
The square root is : 1.414214
himani@Himani:~/Desktop/DS/square$ ./sq_client localhost 12 13
Enter the number whose square root is to be found: 18
The square root is : 4.242641
himani@Himani:~/Desktop/DS/square$ ./sq_client localhost 12 13
Enter the number whose square root is to be found: 100
The square root is : 10.000000
himani@Himani:~/Desktop/DS/square$
```

Server Side

```
himani@Himani: ~/Desktop/DS/square
himani@Himani:~$ cd Desktop/DS/square/
himani@Himani:~/Desktop/DS/square$ gedit square.x
himani@Himani:~/Desktop/DS/square$ gedit sq.x
^C
himani@Himani:~/Desktop/DS/square$ rpcgen -a -C sq.x
himani@Himani:~/Desktop/DS/square$ make -f Makefile.sq
cc -g      -c -o sq_clnt.o sq_clnt.c
cc -g      -c -o sq_client.o sq_client.c
cc -g      -c -o sq_xdr.o sq_xdr.c
cc -g      -o sq_client sq_clnt.o sq_client.o sq_xdr.o -lnsl -lm
cc -g      -c -o sq_svc.o sq_svc.c
cc -g      -c -o sq_server.o sq_server.c
cc -g      -o sq_server sq_svc.o sq_server.o sq_xdr.o -lnsl -lm
himani@Himani:~/Desktop/DS/square$ ./sq_server

Number received is 16.000000

Number received is 2.000000

Number received is 18.000000

Number received is 100.000000

```

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

Source Code:

swap.x

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

program SWAP_PROG
{
    version SWAP_VERS
    {
        intpair SWAP(intpair) = 1;
    } = 1;
} = 0x23451111;
```

swap_client.c

```
#include "swap.h"

void swap_prog_1(char *host,int a ,int b)
```

```

{
    CLIENT *clnt;
    intpair *result_1;
    intpair swap_1_arg;

#ifdef DEBUG
    clnt = clnt_create (host, SWAP_PROG, SWAP_VERS, "udp");
    if (clnt == NULL) {
        clnt_pcreateerror (host);
        exit (1);
    }
#endif /* DEBUG */
    swap_1_arg.a=a;
    swap_1_arg.b=b;
    result_1 = swap_1(&swap_1_arg, clnt);
    if (result_1 == (intpair *) NULL) {
        clnt_perror (clnt, "call failed");
    }
    else{
        swap_1_arg=*result_1;
        printf("The Swapped Numbers are a = %d and b = %d\n", swap_1_arg.a, swap_1_arg.b);
    }
#ifdef DEBUG
    clnt_destroy (clnt);
#endif /* DEBUG */
}

int main (int argc, char *argv[])
{
    char *host;
    int a,b;
    if (argc < 2) {
        printf ("usage: %s server_host\n", argv[0]);
        exit (1);
    }
    host = argv[1];
    printf("Enter the Numbers to beswapped: ");
    scanf("%d %d", &a, &b);
    swap_prog_1 (host,a,b);
    exit (0);
}

```

swap_server.c

```

#include "swap.h"

intpair * swap_1_svc(intpair *argp, struct svc_req *rqstp)
{
    static intpair result;
    int a = argp->a;
    int b = argp->b;
    printf("\nValue of a received is %d\n", a);

```

```
printf("Value of b received is %d\n", b);
result.a = b;
result.b = a;

return &result;
}
```

Output:

Client Side

```
himani@Himani: ~/Desktop/DS/swap
himani@Himani:~/Desktop/DS/swap$ ./swap_client localhost 12 13
Enter the Numbers to be swapped: 3 4
The Swapped Numbers are a = 4 and b = 3
himani@Himani:~/Desktop/DS/swap$ ./swap_client localhost 12 13
Enter the Numbers to be swapped: -5 -1
The Swapped Numbers are a = -1 and b = -5
himani@Himani:~/Desktop/DS/swap$ ./swap_client localhost 12 13
Enter the Numbers to be swapped: 178 182
The Swapped Numbers are a = 182 and b = 178
himani@Himani:~/Desktop/DS/swap$ ./swap_client localhost 12 13
Enter the Numbers to be swapped: 34 89
The Swapped Numbers are a = 89 and b = 34
himani@Himani:~/Desktop/DS/swap$
```

Server Side

```
himani@Himani: ~/Desktop/DS/swap
himani@Himani:~/Desktop/DS/swap$ make -f Makefile.swap
cc -g -c -o swap_xdr.o swap_xdr.c
cc -g -o swap_client swap_clnt.o swap_client.o swap_xdr.o -lnsl
cc -g -c -o swap_svc.o swap_svc.c
cc -g -c -o swap_server.o swap_server.c
cc -g -o swap_server swap_svc.o swap_server.o swap_xdr.o -lnsl
himani@Himani:~/Desktop/DS/swap$ ./swap_server

Value of a received is 3
Value of b received is 4

Value of a received is -5
Value of b received is -1

Value of a received is 178
Value of b received is 182

Value of a received is 34
Value of b received is 89
```

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

Source Code:

arrop.x

```
struct arropair {
    int arr[100];
    int len;
};

struct arpdata {
    int max;
    int min;
    double avg;
};

program ARROPER_PROG
{
    version ARROPER_VERS
    {
        arpdata ARROPER(arropair) = 1;
    } = 1;
} = 0x23451111;
```

arrop_client.c

```
#include "arrop.h"

void arrop_prog_1(char *host,int n,int arr[])
{
    CLIENT *clnt;
    arpdata *result_1;
    arropair arrop_1_arg;
    int i;

#ifdef DEBUG
    clnt = clnt_create (host, ARROPER_PROG, ARROPER_VERS, "udp");
    if (clnt == NULL) {
        clnt_pcreateerror (host);
        exit (1);
    }
#endif /* DEBUG */
    arrop_1_arg.len = n;

    for(i=0;i<n;i++){
        arrop_1_arg.arr[i] = arr[i];
    }

    result_1 = arrop_1(&arrop_1_arg, clnt);
    if (result_1 == (arpdata *) NULL) {
        clnt_perror (clnt, "call failed");
    }else{
```

```

        printf("The max value of array is: %d\n",result_1->max);
        printf("The min value of array is: %d\n",result_1->min);
        printf("The avg value of array is: %.2lf\n",result_1->avg);
    }
#ifdef DEBUG
    clnt_destroy (clnt);
#endif /* DEBUG */
}

int main (int argc, char *argv[])
{
    char *host;
    int n,i;
    int arr[100];

    if (argc < 2) {
        printf ("usage: %s server_host\n", argv[0]);
        exit (1);
    }
    host = argv[1];
    printf("Enter the length of the array (less than 100): ");
    scanf("%d",&n);

    for(i=0;i<n;i++){
        scanf("%d",&arr[i]);
    }

    arropar_prog_1 (host,n,arr);
    exit (0);
}

```

arrop_server.c

```

#include "arrop.h"

arndata * arropar_1_svc(arrpair *argp, struct svc_req *rqstp)
{
    static arndata  result;
    int i = 0;
    int max = argp->arr[0], min = argp->arr[0], total = 0;
    double avg = 0;
    int N = argp->len;

    printf("\nSERVER: Array received\n");

    for(i=0;i<N;i++){
        if(argp->arr[i]>max) max = argp->arr[i];

        if(argp->arr[i]<min) min = argp->arr[i];

        total+=argp->arr[i];
    }
}

```

```

    avg = (double)total/((double)N;

    result.max = max;
    result.min = min;
    result.avg = avg;
    return &result;
}

```

Output:

Client Side

```

himani@Himani: ~/Desktop/DS/array operations
himani@Himani:~/Desktop/DS/array operations$ ./arrop_client localhost 12 13
Enter the length of the array (less than 100): 4
5 2 7 1
The max value of array is: 7
The min value of array is: 1
The avg value of array is: 3.75
himani@Himani:~/Desktop/DS/array operations$ ./arrop_client localhost 12 13
Enter the length of the array (less than 100): 6
2 9 5 6 8 4
The max value of array is: 9
The min value of array is: 2
The avg value of array is: 5.67
himani@Himani:~/Desktop/DS/array operations$ ./arrop_client localhost 12 13
Enter the length of the array (less than 100): 3
6 4 2
The max value of array is: 6
The min value of array is: 2
The avg value of array is: 4.00
himani@Himani:~/Desktop/DS/array operations$ 

```

Server Side

```

himani@Himani: ~/Desktop/DS/array operations
himani@Himani:~/Desktop/DS/array operations$ make -f Makefile.arrop
cc -g      -c -o arrop_clnt.o arrop_clnt.c
cc -g      -c -o arrop_client.o arrop_client.c
cc -g      -c -o arrop_xdr.o arrop_xdr.c
cc -g      -o arrop_client  arrop_clnt.o arrop_client.o arrop_xdr.o -lnsl
cc -g      -o arrop_server  arrop_svc.o arrop_server.o arrop_xdr.o -lnsl
himani@Himani:~/Desktop/DS/array operations$ ./arrop_server

SERVER: Array received

SERVER: Array received

SERVER: Array received

```

7. Compare the given two strings.

Source Code:

comp.x

```
struct strdata {
    char str1[20];
    char str2[20];
};

program COMP_PROG
{
    version COMP_VERS
    {
        int COMP(strdata) = 1;
    } = 1;
} = 0x23451111;
```

comp_client.c

```
#include "comp.h"

void comp_prog_1(char *host, char str1[], char str2[])
{
    CLIENT *clnt;
    int *result_1;
    strdata comp_1_arg;

#ifdef DEBUG
    clnt = clnt_create (host, COMP_PROG, COMP_VERS, "udp");
    if (clnt == NULL) {
        clnt_pcreateerror (host);
        exit (1);
    }
#endif /* DEBUG */
    strcpy(comp_1_arg.str1, str1);
    strcpy(comp_1_arg.str2, str2);
    result_1 = comp_1(&comp_1_arg, clnt);
    if (result_1 == (int *) NULL) {
        clnt_perror (clnt, "call failed");
    } else {
        if (*result_1 == 0) {
            printf("Both the strings are identical.\n");
        } else if (*result_1 < 0) {
            printf("String 2 is greater than String 1.\n");
        } else {
            printf("String 1 is greater than String 2.\n");
        }
    }
}

#ifdef DEBUG
    clnt_destroy (clnt);
#endif /* DEBUG */
}
```

```

int main (int argc, char *argv[])
{
    char *host;
    char str1[20],str2[20];
    if (argc < 2) {
        printf ("usage: %s server_host\n", argv[0]);
        exit (1);
    }
    host = argv[1];

    printf("Enter the first string: ");
    gets(str1);
    printf("Enter the second string: ");
    gets(str2);
    comp_prog_1 (host,str1,str2);
    exit (0);
}

```

comp_server.c

```

#include "comp.h"
#include<string.h>

int * comp_1_svc(strdata *argp, struct svc_req *rqstp)
{
    static int result;
    printf("\nStrings received are %s and %s\n",argp->str1,argp->str2);

    result = strcmp(argp->str1,argp->str2);
    return &result;
}

```

Output:

Client Side

```

himani@Himani:~/Desktop/DS/compare strings$ ./comp_client localhost 12 13
Enter the first string: hello
Enter the second string: hello
Both the strings are identical.
himani@Himani:~/Desktop/DS/compare strings$ ./comp_client localhost 12 13
Enter the first string: hello
Enter the second string: world
String 2 is greater than String 1.
himani@Himani:~/Desktop/DS/compare strings$ ./comp_client localhost 12 13
Enter the first string: world1
Enter the second string: world2
String 2 is greater than String 1.
himani@Himani:~/Desktop/DS/compare strings$

```

Server Side

```
himani@Himani: ~/Desktop/DS/compare strings
himani@Himani:~/Desktop/DS/compare strings$ make -f Makefile.comp
make: Nothing to be done for 'all'.
himani@Himani:~/Desktop/DS/compare strings$ ./comp_server

Strings received are hello and hello

Strings received are hello and world

Strings received are world1 and world2
```

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

Source Code:

subs.x

```
struct strdata {
    char str[20];
    char substr[20];
};

program SUBS_PROG
{
    version SUBS_VERS
    {
        int SUBS(strdata) = 1;
    } = 1;
} = 0x23451111;
```

subs_client.c

```
#include "subs.h"

void subs_prog_1(char *host, char str[], char substr[])
{
    CLIENT *clnt;
    int *result_1;
    strdata subs_1_arg;

#ifdef DEBUG
    clnt = clnt_create (host, SUBS_PROG, SUBS_VERS, "udp");
    if (clnt == NULL) {
        clnt_pcreateerror (host);
        exit (1);
    }
#endif /* DEBUG */
    strcpy(subs_1_arg.str, str);
```

```
strcpy(subs_1_arg.substr,substr);
result_1 = subs_1(&subs_1_arg, clnt);
if (result_1 == (int *) NULL) {
    clnt_perror (clnt, "call failed");
}
else{
    if(*result_1==1){
        printf("It is a Substring!\n");
    }
    else{
        printf("It is not a Substring!\n");
    }
}

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

int main (int argc, char *argv[])
{
    char *host;
    char str[20], substr[20];
    if (argc < 2) {
        printf ("usage: %s server_host\n", argv[0]);
        exit (1);
    }
    host = argv[1];
    printf("Enter the initial string: ");
    gets(str);
    printf("Enter the sub string: ");
    gets(substr);
    subs_prog_1 (host,str,substr);
    exit (0);
}
```

subs_server.c

```
#include "subs.h"
#include<string.h>
int * subs_1_svc(strdata *argp, struct svc_req *rqstp)
{
    static int result=0;
    int N = strlen(argp->str);
    int M = strlen(argp->substr);
    printf("\nString received is ");
    puts(argp->str);
    printf("While Substring received is ");
    puts(argp->substr);
    printf("\n");
    for(int i = 0; i <= N - M; i++){
        int j;
```

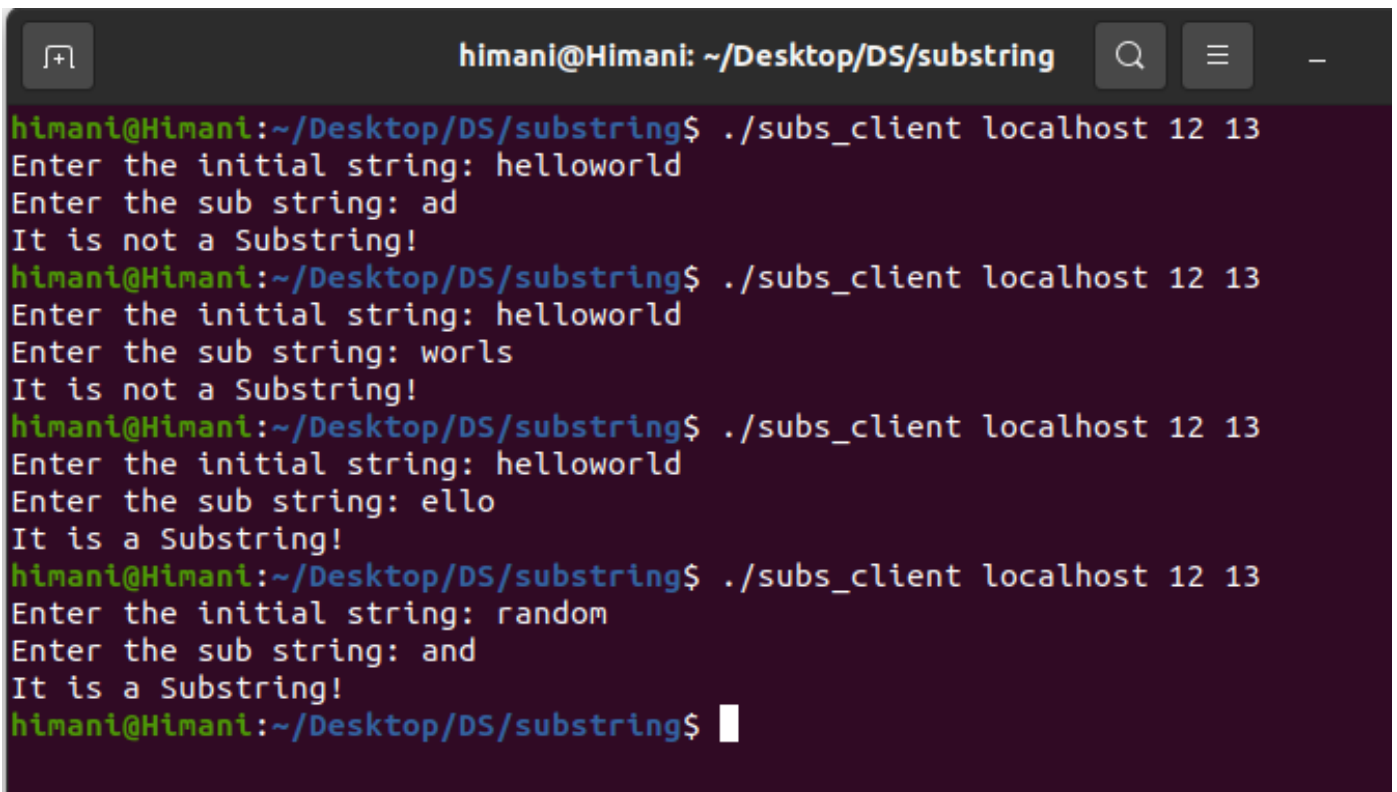
```
    for (j = 0; j < M; j++)
        if (argp->str[i + j] != argp->substr[j]) break;

    if (j == M){
        result = 1;
    }
}

return &result;
}
```

Output:

Client Side



A terminal window titled 'himani@Himani: ~/Desktop/DS/substring' showing the execution of the './subs_client localhost 12 13' command. The program prompts for an initial string and a sub string, then checks if the sub string is a substring of the initial string. The output shows three successful matches: 'ello' in 'helloworld', 'and' in 'random', and 'worls' in 'helloworld'.

```
himani@Himani:~/Desktop/DS/substring$ ./subs_client localhost 12 13
Enter the initial string: helloworld
Enter the sub string: ad
It is not a Substring!
himani@Himani:~/Desktop/DS/substring$ ./subs_client localhost 12 13
Enter the initial string: helloworld
Enter the sub string: worls
It is not a Substring!
himani@Himani:~/Desktop/DS/substring$ ./subs_client localhost 12 13
Enter the initial string: helloworld
Enter the sub string: ello
It is a Substring!
himani@Himani:~/Desktop/DS/substring$ ./subs_client localhost 12 13
Enter the initial string: random
Enter the sub string: and
It is a Substring!
himani@Himani:~/Desktop/DS/substring$
```

Server Side

```
himani@Himani: ~/Desktop/DS/substring
himani@Himani:~/Desktop/DS/substring$ make -f Makefile.subs
make: Nothing to be done for 'all'.
himani@Himani:~/Desktop/DS/substring$ ./subs_server

String received is helloworld
While Substring received is ad

String received is helloworld
While Substring received is worls

String received is helloworld
While Substring received is ello

String received is random
While Substring received is and
```

9. Concatenate the two strings.

Source Code:

concat.x

```
struct strdata {
    char str1[20];
    char str2[20];
};

struct strfin {
    char str3[40];
};

program CONCAT_PROG
{
    version CONCAT_VERS
    {
        strfin CONCAT(strdata) = 1;
    } = 1;
} = 0x23451111;
```

concat_client.c

```
#include "concat.h"
```

```

void concat_prog_1(char *host,char str1[],char str2[])
{
    CLIENT *clnt;
    strfin *result_1;
    strdata concat_1_arg;

#ifdef DEBUG
    clnt = clnt_create (host, CONCAT_PROG, CONCAT_VERS, "udp");
    if (clnt == NULL) {
        clnt_pcreateerror (host);
        exit (1);
    }
#endif /* DEBUG */
    strcpy(concat_1_arg.str1,str1);
    strcpy(concat_1_arg.str2,str2);
    result_1 = concat_1(&concat_1_arg, clnt);
    if (result_1 == (strfin *) NULL) {
        clnt_perror (clnt, "call failed");
    }
    else{
        printf("The conacatenated string is: ");
        puts(result_1->str3);
    }
#ifdef DEBUG
    clnt_destroy (clnt);
#endif /* DEBUG */
}

int main (int argc, char *argv[])
{
    char *host;
    char str1[20], str2[20];
    if (argc < 2) {
        printf ("usage: %s server_host\n", argv[0]);
        exit (1);
    }
    host = argv[1];
    printf("Enter the first string: ");
    gets(str1);
    printf("Enter the second string: ");
    gets(str2);
    concat_prog_1 (host,str1,str2);
    exit (0);
}

```

concat_server.c

```

#include "concat.h"

strfin * concat_1_svc(strdata *argp, struct svc_req *rqstp)
{
    static strfin result;

```

```

printf("\nStrings received are %s and %s\n",argp->str1,argp->str2);

strcpy(result.str3,argp->str1);
strcat(result.str3,argp->str2);

return &result;
}

```

Output:

Client Side

```

himani@Himani: ~/Desktop/DS/concat strings
himani@Himani:~/Desktop/DS/concat strings$ ./concat_client localhost 12 13
Enter the first string: Hello
Enter the second string: World
The concatenated string is: HelloWorld
himani@Himani:~/Desktop/DS/concat strings$ ./concat_client localhost 12 13
Enter the first string: Hi
Enter the second string: Five
The concatenated string is: HiFive
himani@Himani:~/Desktop/DS/concat strings$

```

Server Side

```

himani@Himani: ~/Desktop/DS/concat strings
himani@Himani:~/Desktop/DS/concat strings$ make -f Makefile.concat
make: Nothing to be done for 'all'.
himani@Himani:~/Desktop/DS/concat strings$ ./concat_server

Strings received are Hello and World

Strings received are Hi and Five

```

10. Reverse the elements of an array.

Source Code:

rev.x

```

struct arr1 {
    int arr[100];
    int len;
};

struct arr2 {
    int rev[100];
};

```



```

program REV_PROG
{
    version REV_VERS
    {
        arr2 REV(arr1) = 1;
    } = 1;
} = 0x23451111;

```

rev_client.c

```

#include "rev.h"

void rev_prog_1(char *host, int len, int arr[])
{
    CLIENT *clnt;
    arr2 *result_1;
    arr1 rev_1_arg;
    int i;
#ifdef DEBUG
    clnt = clnt_create (host, REV_PROG, REV_VERS, "udp");
    if (clnt == NULL) {
        clnt_pcreateerror (host);
        exit (1);
    }
#endif /* DEBUG */
    rev_1_arg.len=len;
    for(i=0; i<len; i++){
        rev_1_arg.arr[i]=arr[i];
    }
    result_1 = rev_1(&rev_1_arg, clnt);
    if (result_1 == (arr2 *) NULL) {
        clnt_perror (clnt, "call failed");
    }
    else{
        printf("The reverse of the array is: ");
        for(i=0; i<len; i++){
            printf("%d ", result_1->rev[i]);
        }
        printf("\n");
    }
#ifdef DEBUG
    clnt_destroy (clnt);
#endif /* DEBUG */
}

int main (int argc, char *argv[])
{
    char *host;
    int len,arr[100],i;
    if (argc < 2) {

```

```

        printf ("usage: %s server_host\n", argv[0]);
        exit (1);
    }
    host = argv[1];
    printf("Enter the Array length: ");
    scanf("%d", &len);
    printf("Enter the Array elements: ");
    for(i=0; i<len; i++){
        scanf("%d", &arr[i]);
    }
    rev_prog_1 (host,len,arr);
    exit (0);
}

```

rev_server.c

```

#include "rev.h"

arr2 * rev_1_svc(arr1 *argp, struct svc_req *rqstp)
{
    static arr2  result;
    int i;
    int n = argp->len;
    printf("\nArray received\n");
    for(i=0; i<n; i++){
        result.rev[i]=argp->arr[n-i-1];
    }
    return &result;
}

```

Output:

Client Side

```

himani@Himani: ~/Desktop/DS/reverse array
himani@Himani:~/Desktop/DS/reverse array$ ./rev_client localhost 12 13
Enter the Array length: 5
Enter the Array elements: 5 4 3 2 1
The reverse of the array is: 1 2 3 4 5
himani@Himani:~/Desktop/DS/reverse array$ ./rev_client localhost 12 13
Enter the Array length: 4
Enter the Array elements: 8 7 6 5
The reverse of the array is: 5 6 7 8
himani@Himani:~/Desktop/DS/reverse array$ ./rev_client localhost 12 13
Enter the Array length: 3
Enter the Array elements: 6 5 4
The reverse of the array is: 4 5 6
himani@Himani:~/Desktop/DS/reverse array$

```

Server Side



```
himani@Himani:~/Desktop/DS/reverse array$ make -f Makefile.rev
```

```
rpcgen rev.x
```

```
rpcgen rev.x
```

```
cc -g -c -o rev_clnt.o rev_clnt.c
```

```
cc -g -c -o rev_client.o rev_client.c
```

```
rpcgen rev.x
```

```
cc -g -c -o rev_xdr.o rev_xdr.c
```

```
cc -g -o rev_client rev_clnt.o rev_client.o rev_xdr.o -lnsl
```

```
cc -g -c -o rev_svc.o rev_svc.c
```

```
cc -g -c -o rev_server.o rev_server.c
```

```
cc -g -o rev_server rev_svc.o rev_server.o rev_xdr.o -lnsl
```

```
himani@Himani:~/Desktop/DS/reverse array$ ./rev_server
```

```
Array received
```

```
Array received
```

```
Array received
```