U18CO018

Shekhaliya Shubham

DA

Assignment 6

1. String is palindrome or not.

Code:

```
==> pal.x <==
struct input {
        int n; char
        s[100];
};

program PAL_PROG {
        version PAL_VERS {
                int pal(input)=1;
        }=1;
}=0x12345678;


==> pal_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 "pal.h"

int *
pal_1_svc(input *argp, struct svc_req *rqstp)
{ static int result;

   /*
* insert server code here
    */
        int n = argp->n;


   printf("Received input in server of size %d\n", n);
```

```
    int ans = 1;

    for(int i = 0; i < n; i++) {
        if(argp->s[i] != argp->s[n - 1 - i]) {
            ans = 0;
            break;
        }
    }


    result = ans;
    return &result;
}

==> pal_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 "pal.h"


void pal_prog_1(char *host, int n,
char *s)
{
    CLIENT *clnt; int
    *result_1; input
    pal_1_arg;

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

            pal_1_arg.n = n;
            for(int i = 0; i < n; i++) {
                        pal_1_arg.s[i] = s[i];
            }

    result_1 = pal_1(&pal_1_arg, clnt);
    if (result_1 == (int *) NULL) {
        clnt_perror (clnt, "call failed");
    } else {
        if(*result_1) {
```

```
            printf("String is a palindrome
        \n"); } else { printf("String is not a
            palindrom \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; printf("Enter the size of
            string\n"); scanf("%d",&n);

    char s[100];
    printf("Enter string: \n");
    scanf("%s", &s);
    pal_prog_1 (host, n, s);
}
```

```
root@ubuntu-bionic:/vagrant/assign6/1# ./pal_client localhost
Enter the size of string
3
Enter string:
aba
String is a palindrome
root@ubuntu-bionic:/vagrant/assign6/1# ./pal_client localhost
Enter the size of string
5
Enter string:
abcss
String is not a palindrom
root@ubuntu-bionic:/vagrant/assign6/1# ./pal_client localhost
Enter the size of string
6
Enter string:
aabbaa
String is a palindrome
root@ubuntu-bionic:/vagrant/assign6/1# ./pal_client localhost
Enter the size of string
6
Enter string:
abcsfd
String is not a palindrom
```

```
root@ubuntu-bionic:/vagrant/assign6/1# ./pal_server
Received input in server of size 3
Received input in server of size 5
Received input in server of size 6
Received input in server of size 6
```

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

Code:

```
==> leap.x <==
struct input {
        int year;
};

program LEAP_PROG {
        version LEAP_VERS {
                int leap(input)=1;
        }=1;
}=0x12345678;
```

==> leap_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 "leap.h"

int *
leap_1_svc(input *argp, struct svc_req
*rqstp) { static int result;

        /*
* insert server code here
         */

        printf("Server received input %d\n", argp->year);

        int year = argp->year;

        if (year % 400 == 0) {
          result = 1;
  }
  else if (year % 100 == 0) {
          result = 0;
  } else if (year % 4 ==
  0) { result = 1;
  } else { result =
  0;
  }
        return &result;
}
```

==> leap_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 "leap.h"


void leap_prog_1(char *host, int
year)
{
```

```
        CLIENT *clnt; int
        *result_1; input
        leap_1_arg;

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

        leap_1_arg.year = year; result_1 =
        leap_1(&leap_1_arg, clnt); if
        (result_1 == (int *) NULL) {
        clnt_perror (clnt, "call failed");
        } else { if (*result_1) { printf("Leap
        Year\n");
                } else { printf("Non Leap Year
                        \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 year; printf("Enter input
        year: \n");
        scanf("%d",&year);

        leap_prog_1 (host,year);
exit (0);
}
```

```
root@ubuntu-bionic:/vagrant/assign6/2# ./leap_client localhost
Enter input year:
2000
Leap Year
root@ubuntu-bionic:/vagrant/assign6/2# ./leap_client localhost
Enter input year:
2001
Non Leap Year
root@ubuntu-bionic:/vagrant/assign6/2# ./leap_client localhost
Enter input year:
201
Non Leap Year
root@ubuntu-bionic:/vagrant/assign6/2# ./leap_client localhost
Enter input year:
2016
Leap Year
root@ubuntu-bionic:/vagrant/assign6/2# ./leap_client localhost
Enter input year:
2020
Leap Year
root@ubuntu-bionic:/vagrant/assign6/2# ./leap_client localhost
Enter input year:
2100
Non Leap Year
```

```
root@ubuntu-bionic:/vagrant/assign6/2# ./leap_server
Server received input 2000
Server received input 2001
Server received input 201
Server received input 2016
Server received input 2020
Server received input 2100
```

3. Find out the GCD of a given number.

Code: ==>

```
gcd.x <==
struct input {
        int a;
    int b;
};

program GCD_PROG {
        version GCD_VERS {
                int gcd(input)=1;
        }=1;
}=0x12345678;


==> gcd_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 "gcd.h"

int *
gcd_1_svc(input *argp, struct svc_req
*rqstp) { static int result;

        /*
* insert server code here
         */
        int a = argp->a, b = argp->b;

        printf("Received input (%d,%d)", a, b);

        int g = 1;

        for(int i = 1; i <= a; i++) { if(a % i == 0
                && b % i == 0) g = i;
        }

        result = g;
        return &result;
}
```

==> gcd_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 "gcd.h"


void
gcd_prog_1(char *host, int a, int b)
{
        CLIENT *clnt; int
        *result_1; input
        gcd_1_arg;

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

```
	gcd_1_arg.a = a;
	gcd_1_arg.b = b;

	result_1 = gcd_1(&gcd_1_arg, clnt);
	if (result_1 == (int *) NULL) {
	clnt_perror (clnt, "call failed");
	} else { printf("Result GCD = %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 a, b;

	printf("Enter number 1: \n");
	scanf("%d", &a); printf("Enter number
	2: \n"); scanf("%d", &b);

	gcd_prog_1 (host, a, b);
exit (0);
}
```

```
Enter number 1:
2
Enter number 2:
4
Result GCD = 2
root@ubuntu-bionic:/vagrant/assign6/3# ./gcd_client localhost
Enter number 1:
4
Enter number 2:
15
Result GCD = 1
root@ubuntu-bionic:/vagrant/assign6/3# ./gcd_client localhost
Enter number 1:
15
Enter number 2:
39
Result GCD = 3
```

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

```
==> sqroot.x <==
struct input {
float a;
};

program SQROOT_PROG {
        version SQROOT_VERS {
                float sqroot(input)=1;
        }=1;
}=0x12345678;


==> sqroot_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 <math.h>
#include "sqroot.h"

float * sqroot_1_svc(input *argp, struct svc_req
*rqstp)
{ static float result;

        /*
* insert server code here
         */
```

```
        float n = argp->a;
        printf("Received input number %f\n", n);

        result = sqrt(n);
        return &result;
}

==> sqroot_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 "sqroot.h"


void sqroot_prog_1(char *host,
float n)
{
        CLIENT *clnt; float
        *result_1; input
        sqroot_1_arg;

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

        sqroot_1_arg.a = n; result_1 =
        sqroot_1(&sqroot_1_arg, clnt); if
        (result_1 == (float *) NULL) { clnt_perror
        (clnt, "call failed");
        } else { printf("The square root of %f is %f \n", 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];

        float num; printf("Enter the
         number: \n"); scanf("%f",
         &num);

        sqroot_prog_1 (host,num);
exit (0);
}
```

```
root@ubuntu-bionic:/vagrant/assign6/4# ./sqroot_server
Received input number 23.000000
Received input number 49.000000
Received input number 441.000000
Received input number 432.000000
```

```
root@ubuntu-bionic:/vagrant/assign6/4# ./sqroot_client localhost
Enter the number:
23
The square root of 23.000000 is 4.795832
root@ubuntu-bionic:/vagrant/assign6/4# ./sqroot_client localhost
Enter the number:
49
The square root of 49.000000 is 7.000000
root@ubuntu-bionic:/vagrant/assign6/4# ./sqroot_client localhost
Enter the number:
441
The square root of 441.000000 is 21.000000
root@ubuntu-bionic:/vagrant/assign6/4# ./sqroot_client localhost
Enter the number:
432
The square root of 432.000000 is 20.784609
```

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

```
==> swap.x <==
struct input {
        int a;
    int b;
};

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

```
program SWAP_PROG {
        version SWAP_VERS {
      output SWAP(input)=1;
        }=1;
}=0x12345678;
==> swap_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 "swap.h"
output *
swap_1_svc(input *argp, struct svc_req *rqstp)
{ static output result;

        /*
* insert server code here
         */
        int a= argp->a, b = argp->b;
        a=a+b;//a=30 (10+20) b=a
        b;//b=10 (30-20) a=a-b;//a=20
        (30-10)

        result.a = a;
        result.b = b;

        return &result;
}

==> swap_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 "swap.h"


void swap_prog_1(char *host,int a,
int b)
{
        CLIENT *clnt;
        output *result_1;
        input swap_1_arg;
```

```
#ifndef 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 == (output *) NULL) {
        clnt_perror (clnt, "call failed");
        } else { printf("After swapping \n");
                printf("a = %d\n", result_1-
                >a); printf("b = %d\n",
                result_1->b);
        }
#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 a, b; printf("Enter
        two numbers to swap \n"); scanf("%d
        %d", &a, &b); swap_prog_1 (host,a,b);

exit (0);
}
```

```
root@ubuntu-bionic:/vagrant/assign6/5# ./swap_client localhost
Enter two numbers to swap
3 4
After swapping
a = 4
b = 3
```

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

```
==> stats.x <==
struct input{ int
n; int arr[100];
};
struct output{
    int mx;
    int mn;
    float
    avg;
};

program STATS_PROG{
   version STATS_VERS{
       output stats(input)=1;
   }=1;
}=0x12345678;
==> stats_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 "stats.h"

output * stats_1_svc(input *argp, struct
svc_req *rqstp)
{ static output result;
          /*
* insert server code here
           */

          int n = argp->n; int *arr = argp->arr;

          printf("Received array of size %d\n",

          n);

          int mx = arr[0], mn = arr[0];
          float avg = arr[0];

          for(int i = 0; i < n; i++) { if(arr[i]
                    > mx) mx = arr[i]; if(arr[i]
                    < mn) mn = arr[i]; avg
                    += arr[i];
          }
          avg /= n;
          result.mx = mx;
          result.mn = mn;
```

```
	result.avg = avg;
	return &result;
}

==> stats_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 "stats.h"


void stats_prog_1(char *host, int n, int
*arr)
{
	CLIENT *clnt;
	output *result_1;
	input
	stats_1_arg;

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

	stats_1_arg.n = n;
	for(int i = 0; i < n; i++) {
		stats_1_arg.arr[i] = arr[i];
	}
	result_1 = stats_1(&stats_1_arg, clnt);
	if (result_1 == (output *) NULL) {
	clnt_perror (clnt, "call failed");
	} else { printf("MAX = %d\n", result_1->mx);
		printf("MIN = %d\n", result_1->mn);
		printf("AVG = %f\n", result_1->avg);
	}
#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 the number of

        elements: \n"); scanf("%d", &n); int

        arr[100]; printf("Enter %d elements: \n",

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

                scanf("%d", &arr[i]);
        }

        stats_prog_1 (host, n, arr);
exit (0);
}
```

```
root@ubuntu-bionic:/vagrant/assign6/6# ./stats_client localhost
Enter the number of elements:
5
Enter 5 elements:
1
2
3
4
5
MAX = 5
MIN = 1
AVG = 3.200000
root@ubuntu-bionic:/vagrant/assign6/6# ./stats_client localhost
Enter the number of elements:
6
Enter 6 elements:
12
324
123
442
6
763
MAX = 763
MIN = 6
AVG = 280.333344
root@ubuntu-bionic:/vagrant/assign6/6#
```

7. Compare the given two strings.

```
==> cmp.x <==
struct input{
   int n1; int
   n2; char
   s1[50]; char
```

```
    s2[50];
};

program CMP_PROG{
    version CMP_VERS{
    int cmp(input)=1;
    }=1;
}=0x12345678;
==> cmp_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<string.h>
#include "cmp.h"

int *
cmp_1_svc(input *argp, struct svc_req *rqstp)
{ static int result;

        /*
 * insert server code here
         */
        printf("strings received of size %d %d", argp->n1, argp-

        >n2); printf("%s %s\n", argp->s1, argp->s2); int n1 =

        argp- >n1, n2 = argp->n2;

        int l = n1; if(n1 >

        n2) l = n2; char

        s1[50], s2[50]; int

        ans = 0;


        for(int i = 0; i < l; i++) { char s1 = argp->s1[i],
                s2 = argp->s2[i]; if(s1 > s2) { ans = 1;
                break;
                } else if(s1 < s2) {
                        ans = -1;
                        break;
                }
        }
        if(ans == 0 && n1 != n2) {
                if(n1 < n2) ans = -1;
                else ans = 1;
```

```
        }

        printf("%d \n",
        ans); result = ans;
        return &result;
}

==> cmp_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 "cmp.h"


void cmp_prog_1(char *host, int n1, int n2, char *s1,
char *s2)
{
        CLIENT *clnt; int
        *result_1; input
        cmp_1_arg;

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

        cmp_1_arg.n1 = n1;
        cmp_1_arg.n2 = n2;

        for(int i = 0; i < n1; i++) {
                cmp_1_arg.s1[i] = s1[i];
        }

        for(int i = 0; i < n2; i++) {
                cmp_1_arg.s2[i] = s2[i];
        }

        result_1 = cmp_1(&cmp_1_arg, clnt);
        if (result_1 == (int *) NULL) {
        clnt_perror (clnt, "call failed");
        } else { int ans =

                *result_1;
```

```
                printf("%d\n", ans); if( ans > 0) { printf("String %s is
                larger than %s \n", s1, s2);
                } else if(ans == 0) { printf("String %s is equal to
                        %s \n", s1, s2);
                } else { printf("String %s is smaller than %s \n", s1,
                        s2);
                }
        }
#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 n1, n2; char
        s1[50], s2[50];

        printf("Enter the size of string 1 \n");
        scanf("%d", &n1);

        printf("Enter the size of string 2 \n");
        scanf("%d", &n2);
        printf("Enter string 1 \n");
        scanf("%s", s1);
        printf("Enter string 2 \n");
        scanf("%s", s2);

        cmp_prog_1 (host, n1, n2, s1, s2);
exit (0);
}
```

```
root@ubuntu-bionic:/vagrant/assign6/7# ./cmp_client localhost
Enter the size of string 1
5
Enter the size of string 2
5
Enter string 1
abcde
Enter string 2
abcde
0
String abcde is equal to abcde
root@ubuntu-bionic:/vagrant/assign6/7# ./cmp_client localhost
Enter the size of string 1
4
Enter the size of string 2
3
Enter string 1
abcd
Enter string 2
abc
1
String abcd is larger than abc
root@ubuntu-bionic:/vagrant/assign6/7# ./cmp_client localhost
Enter the size of string 1
5
Enter the size of string 2
5
Enter string 1
abcde
Enter string 2
effas
-1
String abcde is smaller than effas
root@ubuntu-bionic:/vagrant/assign6/7#
```

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

```
==> cmp.x <==
struct input{
    int n1; int
    n2; char
    s1[50];
    char
    s2[50];
};

program CMP_PROG{
   version CMP_VERS{
   int cmp(input)=1;
   }=1;
}=0x12345678;
==> cmp_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 "cmp.h"
#include<stdlib.h>
#include<string.h>

int *
cmp_1_svc(input *argp, struct svc_req *rqstp)
{ static int result;

        /*
* insert server code here
         */
        printf("strings received of size %d %d \n", argp->n1, argp->n2);


        int n1 = argp->n1, n2 = argp->n2;

        char *s1 = (char *)malloc(n1 * sizeof(char *));
        char *s2 = (char *)malloc(n2 * sizeof(char *));

        for(int i = 0; i < n1; i++) {
                s1[i] = argp->s1[i];
        }

        for(int i = 0; i < n2; i++) {
                s2[i] = argp->s2[i];
        } printf("%s %s \n", s1 ,

        s2);

        int ans = 0; for(int i = 0; i + n2 -
        1 < n1; i++) { int f = 1;

                for(int j = 0; j < n2; j++) { if(s1[i+j]
                        == s2[j] ) continue; f = 0;
                        break;
                }
                if(f) {
                        ans = 1;
                        break;
                }
        }

        result = ans;
        return &result;
}

==> cmp_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 "cmp.h"


void cmp_prog_1(char *host, int n1, int n2, char *s1,
char *s2)
{
        CLIENT *clnt; int
        *result_1; input
        cmp_1_arg;

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

        cmp_1_arg.n1 = n1;
        cmp_1_arg.n2 = n2;

        for(int i = 0; i < n1; i++) {
                cmp_1_arg.s1[i] = s1[i];
        }

        for(int i = 0; i < n2; i++) {
                cmp_1_arg.s2[i] = s2[i];
        }

        result_1 = cmp_1(&cmp_1_arg, clnt);
        if (result_1 == (int *) NULL) {
        clnt_perror (clnt, "call failed");
        } else { int ans = *result_1; printf("%d\n", ans); if(ans == 1) {
                printf("String %s is substring of %s \n", s2, s1);
                } else { printf("String %s is not a substring of %s \n", s2,
                s1); }
        }
#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 n1, n2; char
        s1[50], s2[50];

        printf("Enter the size of string \n");
        scanf("%d", &n1);

        printf("Enter the size of sub string \n");
        scanf("%d", &n2);

        printf("Enter string \n");
        scanf("%s", s1);
        printf("Enter sub string \n");
        scanf("%s", s2);

        cmp_prog_1 (host, n1, n2, s1, s2);
exit (0); }
```

```
String abc is not a substring of abcde
root@ubuntu-bionic:/vagrant/assign6/8# ./cmp_client localhost
Enter the size of string
5
Enter the size of sub string
3
Enter string
abcde
Enter sub string
bcd
1
String bcd is substring of abcde
root@ubuntu-bionic:/vagrant/assign6/8# ./cmp_client localhost
Enter the size of string
5
Enter the size of sub string
3
Enter string
abcde
Enter sub string
cdb
0
String cdb is not a substring of abcde
root@ubuntu-bionic:/vagrant/assign6/8#
```

9. Concatenate the two strings.

```
==> cmp.x <==
struct input{ int
n1; int n2; char
s1[50]; char
s2[50];
};

struct output{
    char s[100];
};

program CMP_PROG{
   version CMP_VERS{
       output cmp(input)=1;
   }=1;
}=0x12345678;
==> cmp_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 "cmp.h"

output * cmp_1_svc(input *argp, struct
svc_req *rqstp)
{ static output result;

          /*
* insert server code here
           */

                    printf("strings received of size %d %d \n", argp->n1, argp->n2);


          int n1 = argp->n1, n2 = argp->n2;

          for(int i = 0; i < n1 + n2; i++) { if(i >= n1)
                    result.s[i] = argp->s2[i - n1];
                    else result.s[i] = argp->s1[i];
          }

          return &result;
}

==> cmp_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 "cmp.h"


void
cmp_prog_1(char *host, int n1, int n2, char *s1, char *s2)
{
	CLIENT *clnt;
	output *result_1;
	input cmp_1_arg;
#ifndef DEBUG clnt = clnt_create (host, CMP_PROG,
	CMP_VERS, "udp"); if (clnt == NULL) { clnt_pcreateerror
	(host); exit (1);
	}
#endif /* DEBUG */

	cmp_1_arg.n1 = n1;
	cmp_1_arg.n2 = n2;

	for(int i = 0; i < n1; i++) {
		cmp_1_arg.s1[i] = s1[i];
	}

	for(int i = 0; i < n2; i++) {
		cmp_1_arg.s2[i] = s2[i];
	}

	result_1 = cmp_1(&cmp_1_arg, clnt);
	if (result_1 == (output *) NULL) {
	clnt_perror (clnt, "call failed");
	} else { printf("Combined string %s \n", result_1-
		>s);
	}
#ifndef DEBUG
	clnt_destroy (clnt);
#endif /* DEBUG */
}


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

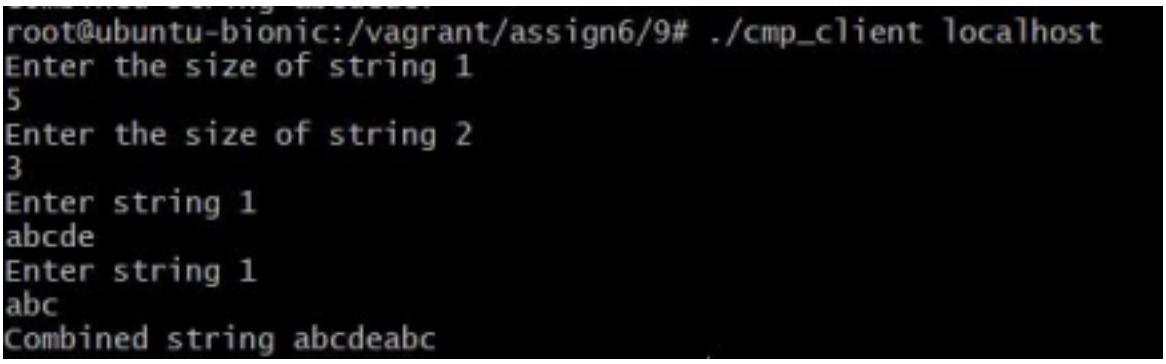
```
        if (argc < 2) { printf ("usage: %s server_host\n",
                argv[0]); exit (1);
        }
        host = argv[1];


        int n1, n2; char
        s1[50], s2[50];
        printf("Enter the size of string 1\n");
        scanf("%d", &n1);

        printf("Enter the size of string 2 \n");
        scanf("%d", &n2);

        printf("Enter string 1\n");
        scanf("%s", s1);
        printf("Enter string 1\n");
        scanf("%s", s2);

        cmp_prog_1 (host, n1, n2, s1, s2);
exit (0);
}
```



10. Reverse the elements of an array.

```
==> cmp.x <==
struct input{ int
n; int a[100];
};

struct output{
    int a[100];
};

program CMP_PROG{
```

```
   version CMP_VERS{
       output cmp(input)=1;
   }=1;
}=0x12345678;
==> cmp_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 "cmp.h"

output * cmp_1_svc(input *argp, struct
svc_req *rqstp)
{ static output result;
        /*
* insert server code here
         */

        int n = argp->n; printf("Received array
        of size %d \n", n);

        for(int i = 0; i < n; i++) { result.a[i] =
                argp->a[n - 1 - i];
        }

        return &result;
}

==> cmp_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 "cmp.h"


void cmp_prog_1(char *host, int n,
int *a)
{
        CLIENT *clnt;
        output *result_1;
        input cmp_1_arg;
#ifndef DEBUG clnt = clnt_create (host, CMP_PROG,
        CMP_VERS, "udp"); if (clnt == NULL) { clnt_pcreateerror
        (host); exit (1);
        }
```

```
#endif /* DEBUG */

        cmp_1_arg.n = n;
        for(int i = 0; i < n; i++) {
                cmp_1_arg.a[i] = a[i];
        }
        result_1 = cmp_1(&cmp_1_arg, clnt);
        if (result_1 == (output *) NULL) {
        clnt_perror (clnt, "call failed");
        } else { printf("Reversed array: \n"); for(int
                i = 0; i < n; i++) { printf("%d ",
                result_1->a[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; printf("Enter the
        number of elements \n"); scanf("%d",
        &n); int a[100]; printf("Enter %d
        elements", n); for(int i = 0; i < n; i++) {
        scanf("%d",&a[i]);
        }
        cmp_prog_1 (host,n,a);
exit (0);
}
```

```
root@ubuntu-bionic:/vagrant/assign6/10# ./cmp_client localhost
Enter the number of elements
5
Enter 5 elements 1 2 3 4 5
Reversed array:
5 4 3 2 1
```