DISTRIBUTED SYSTEMS

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
UNIVERSITY OF WEST ATTICA

DEPARTMENT OF INFORMATION AND COMPUTER ENGINEERING

PROJECT 1
RPC

STUDENT DETAILS

NAME: ATHANASIOU VASILEIOS EVANGELOS
STUDENT ID: 19390005
STUDENT SEMESTER: 8th
STUDENT STATUS: UNDERGRADUATE
STUDY PROGRAM: PADA
LABORATORY DEPARTMENT: ST5-A THURSDAY 12:00 – 14:00
LABORATORY INSTRUCTOR: DOKA AIKATERINI
DELIVERY DATE: 1/5/2023

STUDENT PHOTO:

CONTENTS

Brief Report

rpc _ server . c

O concurrent rpc server receives the following data

- A real number r
- Two integer vectors X, Y of length n

and performs the following 3 calculations :

1. The inner product of the two vectors X * Y (returns an integer)
2. The average of each vector: E x , E y (returns an array of 2 real numbers)
3. The product r * (X + Y) (returns a vector of real numbers of length n)

It returns the results to rpc client (rpc _ client . c) which called the corresponding rpc routine.

rpc_client.c

The rpc client connects via TCP AF_INET socket with a socket client (socket_client.c) and it works in parallel and as a socket server. It receives the data from the socket client , calls the corresponding rpc routine according to the option sent to it, receives the results from the rpc server and finally sends them to the socket client .

socket_client.c

The socket client does application for login via TCP AF_INET socket with the socket server that it works in parallel and as rpc client (rpc_client.c). Enters the data, sends it to socker server , which calls the corresponding rpc routine from an rpc server (rpc _ server . c). Finally, the socket client receives the results from the socket server and prints them to the output .

Create the rpc files and compile commands

To create the rpc files we defined the required interface file (' rpc . x ').

socket_client.c | rpc.x | rpc.h

```
1  struct inputData {
2      int X<>;
3      int Y<>;
4      double r;
5  };
6
7  struct avgArray {
8      double arr[2];
9  };
10
11 struct productVector {
12     double vec<>;
13 };
14
15 program MENU_PROG {
16         version MENU_VER {
17             int INNERPRODUCT(inputData) = 1;
18             avgArray AVERAGES(inputData) = 2;
19             productVector PRODUCT(inputData) = 3;
20         } = 1;
21 } = 0x23451111;
22
23
24
```

Figure 1. The interface file " rpc . x " for rpc communication

struct inputData

inputData " structure contains the input data structures of all 3 rpc routines, i.e., the vector of integers X with its size (X < > makes a structure with 2 members, the size of X and the vector X), the vector of integers Y with its magnitude (the same applies to Y < >) and the actual value r .

struct avgArray

The avgArray structure contains the data structure that the 2nd will ^{return} rpc routine (average of each vector), that is, a static array of 2 real numbers whose 1st ^{position} will contain the average value of the vector X and the 2nd ^{position} will contain the average value of the vector Y .

struct productVector

The " productVector " structure contains the data structure that the 3rd will ^{return} rpc routine (the product r * (X + Y)), i.e., a dynamic array of N reals where each position will contain the product r * (X [i] + Y [i]), where i is a position index of an array.

program MENU _ PROG

This program contains the version MENU _ VER which contains the rpc routines that rpc client can call rpc server to execute.

The command to create ready-made program templates for rpc client and rpc server is as follows

```
rpcgen -a -C rpc.x
```

With her implementation her command were created the files rpc.h (Figure 2), Makefile (Figure 3), rpc_client.c, rpc_server.c, rpc_clnt.c, rpc_svc.c and rpc_xdr.c


```
/*
 * Please do not edit this file.
 * It was generated using rpcgen.
 */

#ifndef _RPC_H_RPCGEN
#define _RPC_H_RPCGEN

#include <rpc/rpc.h>


#ifdef __cplusplus
extern "C" {
#endif


struct inputData {
    struct {
        u_int X_len;
        int *X_val;
    } X;
    struct {
        u_int Y_len;
        int *Y_val;
    } Y;
    double r;
};
typedef struct inputData inputData;

struct avgArray {
    double arr[2];
};
typedef struct avgArray avgArray;

struct productVector {
    struct {
        u_int vec_len;
        double *vec_val;
    } vec;
};
typedef struct productVector productVector;

#define MENU_PROG 0x23451111
#define MENU_VER 1

#if defined(__STDC__) || defined(__cplusplus)
#define INNERPRODUCT 1
extern  int * innerproduct_1(inputData *, CLIENT *);
```

Figure 2. The rpc . h file

socket_client.c | rpc.x | rpc.h | rpc_client.c | rpc_server.c | Makefile

```
# This is a template Makefile generated by rpcgen

# Parameters

CLIENT = rpc_client
SERVER = rpc_server

SOURCES_CLNT.c =
SOURCES_CLNT.h =
SOURCES_SVC.c =
SOURCES_SVC.h =
SOURCES.x = rpc.x

TARGETS_SVC.c = rpc_svc.c rpc_server.c rpc_xdr.c
TARGETS_CLNT.c = rpc_clnt.c rpc_client.c rpc_xdr.c
TARGETS = rpc.h rpc_xdr.c rpc_clnt.c rpc_svc.c rpc_client.c rpc_server.c

OBJECTS_CLNT = $(SOURCES_CLNT.c:%.c=%.o) $(TARGETS_CLNT.c:%.c=%.o)
OBJECTS_SVC = $(SOURCES_SVC.c:%.c=%.o) $(TARGETS_SVC.c:%.c=%.o)
# Compiler flags

CFLAGS += -g -DRPC_SVC_FG -I/usr/include/tirpc
LDLIBS += -lnsl -ltirpc
RPCGENFLAGS = -C

# Targets

all : $(CLIENT) $(SERVER)

$(TARGETS) : $(SOURCES.x)
	rpcgen $(RPCGENFLAGS) $(SOURCES.x)

$(OBJECTS_CLNT) : $(SOURCES_CLNT.c) $(SOURCES_CLNT.h) $(TARGETS_CLNT.c)

$(OBJECTS_SVC) : $(SOURCES_SVC.c) $(SOURCES_SVC.h) $(TARGETS_SVC.c)

$(CLIENT) : $(OBJECTS_CLNT)
	$(LINK.c) -o $(CLIENT) $(OBJECTS_CLNT) $(LDLIBS)

$(SERVER) : $(OBJECTS_SVC)
	$(LINK.c) -o $(SERVER) $(OBJECTS_SVC) $(LDLIBS)

clean:
	$(RM) core $(TARGETS) $(OBJECTS_CLNT) $(OBJECTS_SVC) $(CLIENT) $(SERVER)
```

Figure 3. The Makefile file

socket code client is

```
gcc socket_client.c -o socket_client
```

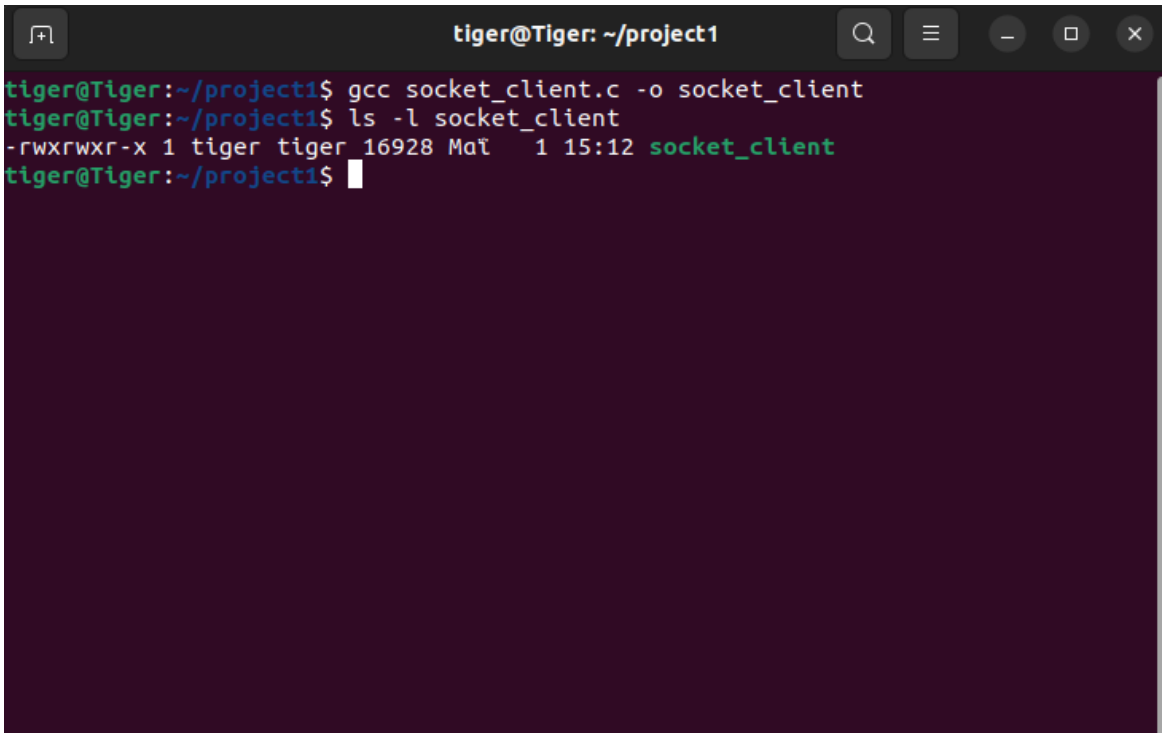


Figure 4. Compiling socket _ client . c

The compile command for rpc _*. c files are

```
make
```

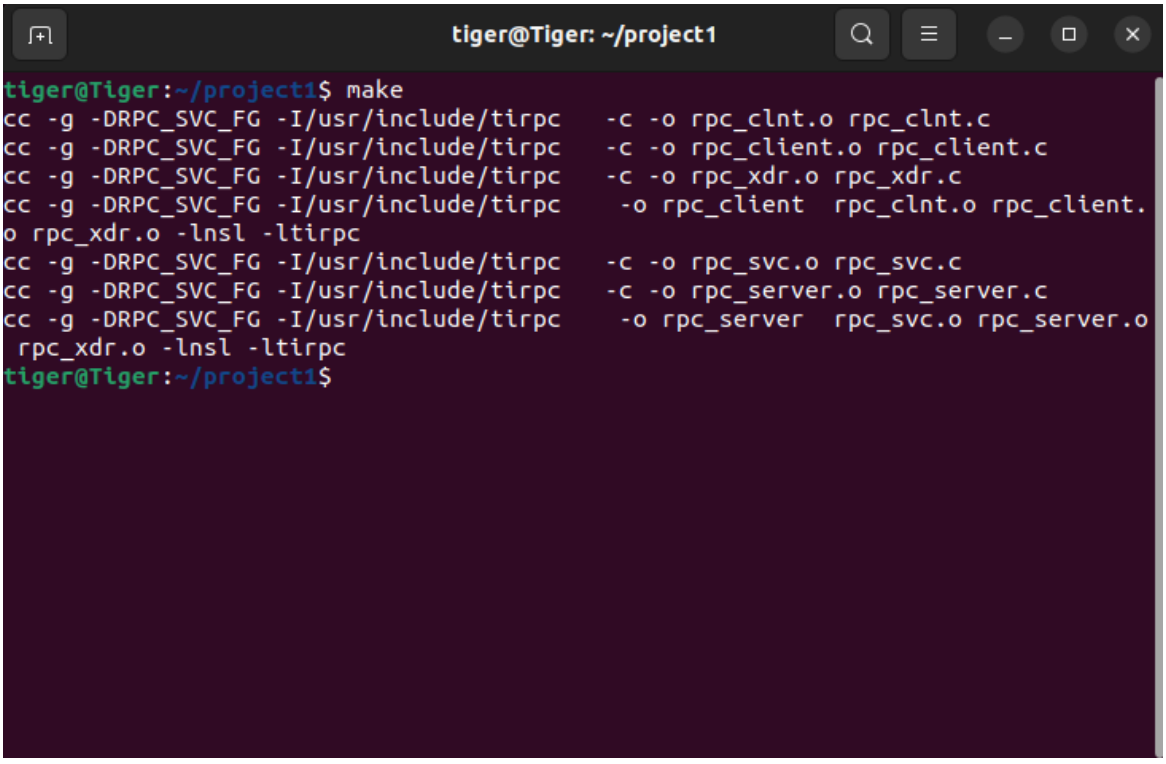
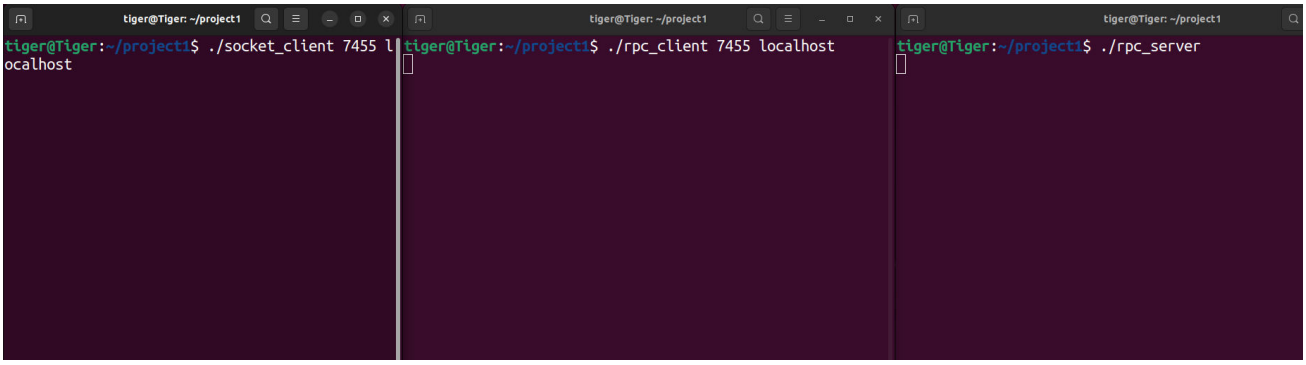


Figure 5. Compilation of rpc _*. c files

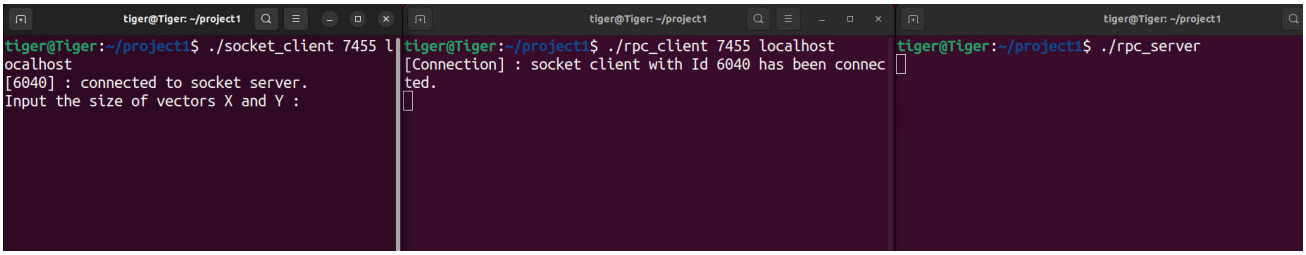
Execution of the programs

Socket clients: 1

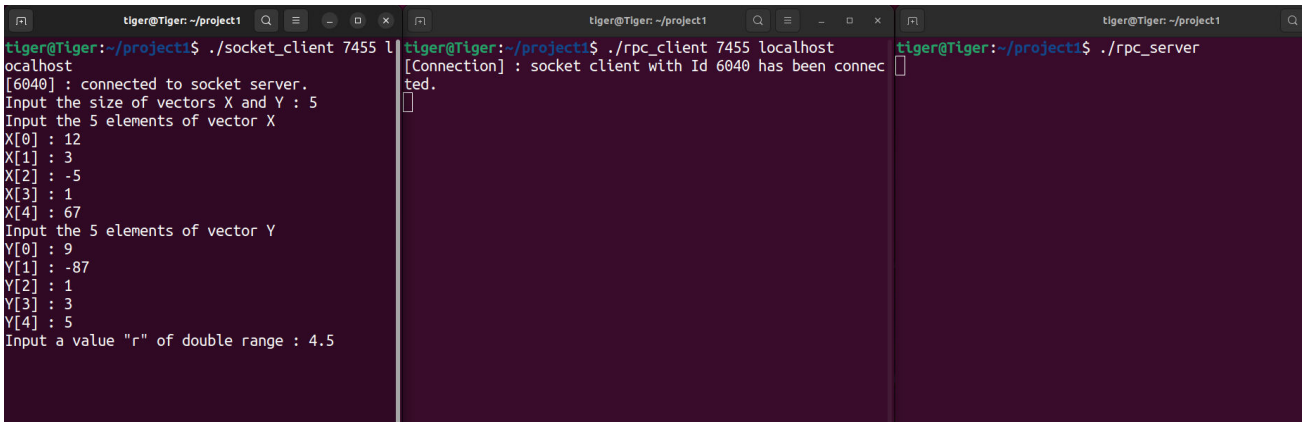
Servers On



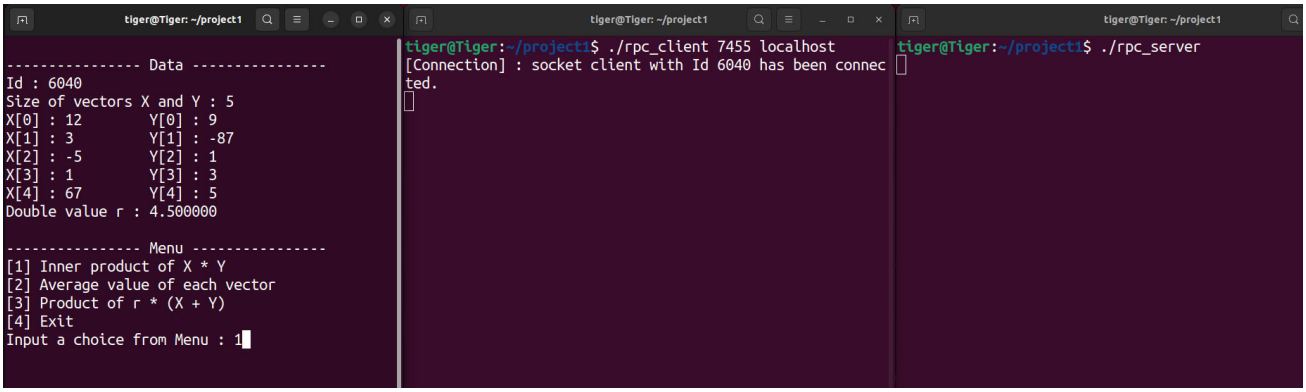
Connection



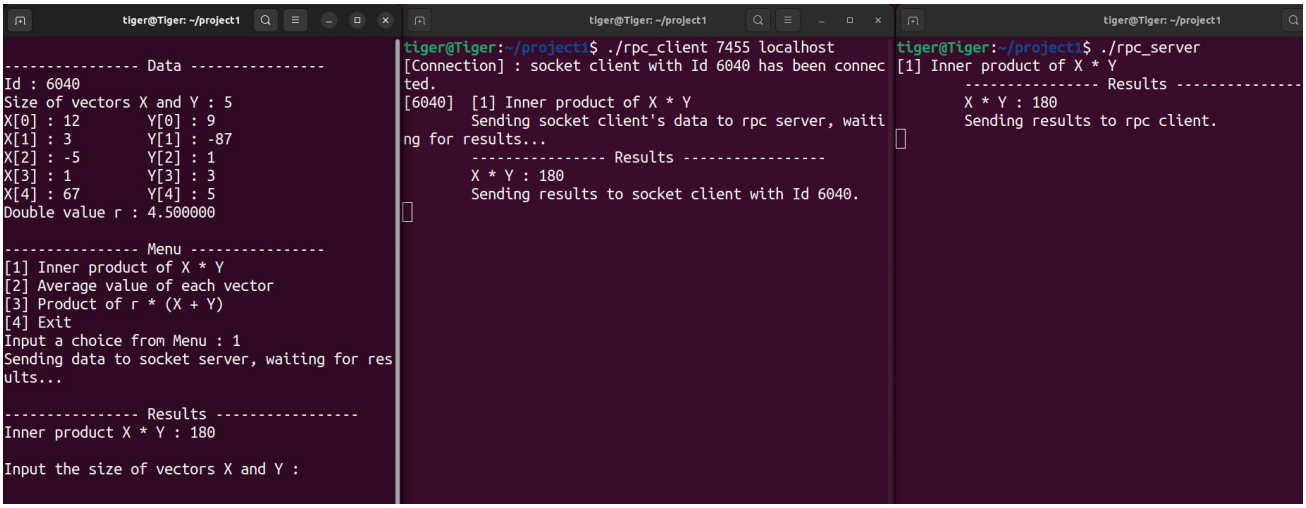
Input Data



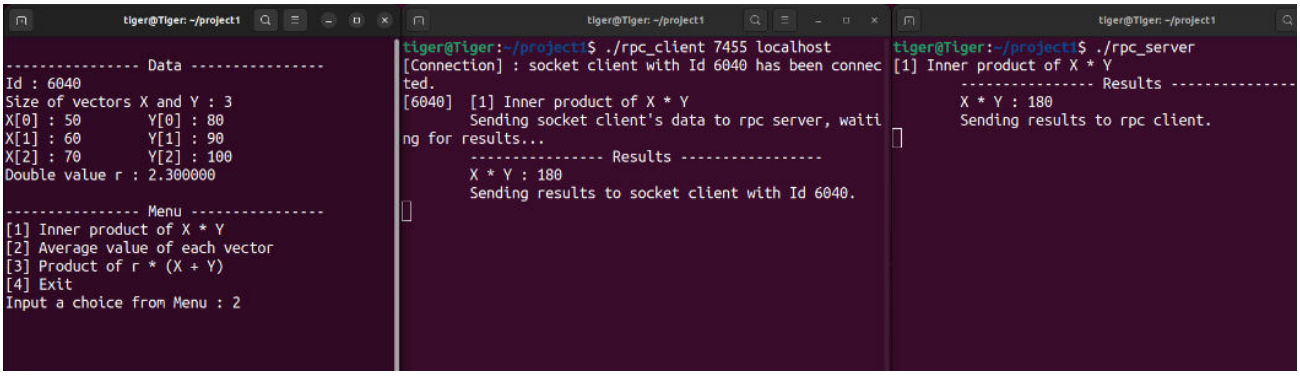
1st Choice



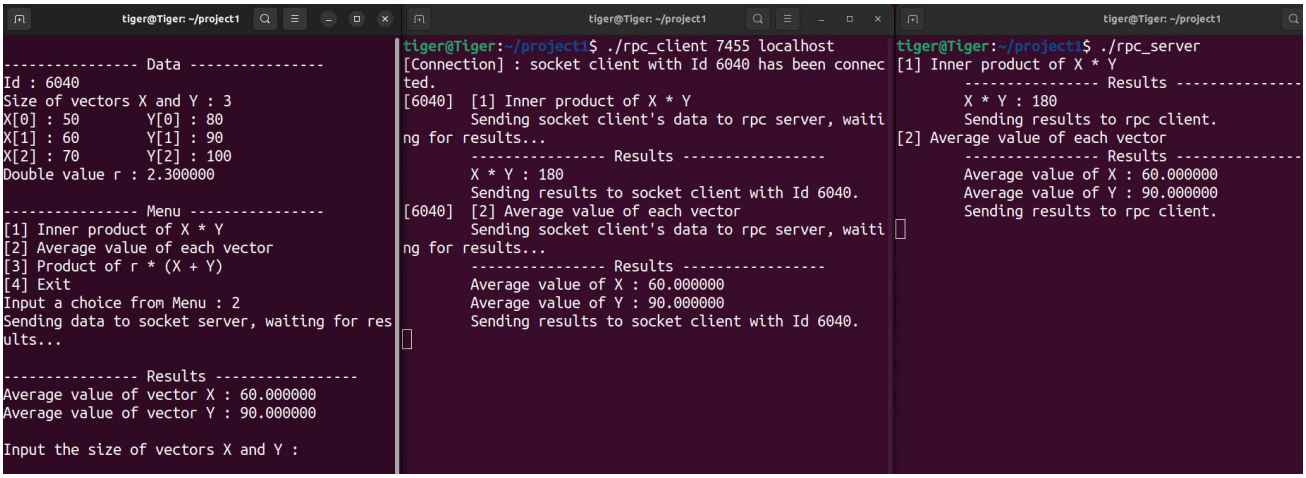
1st Choice Results



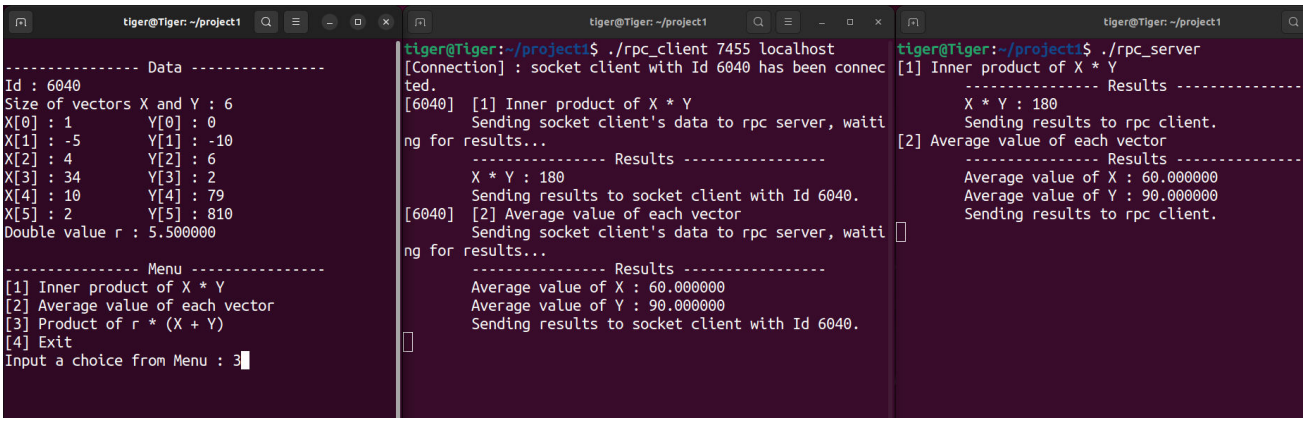
2nd Choice



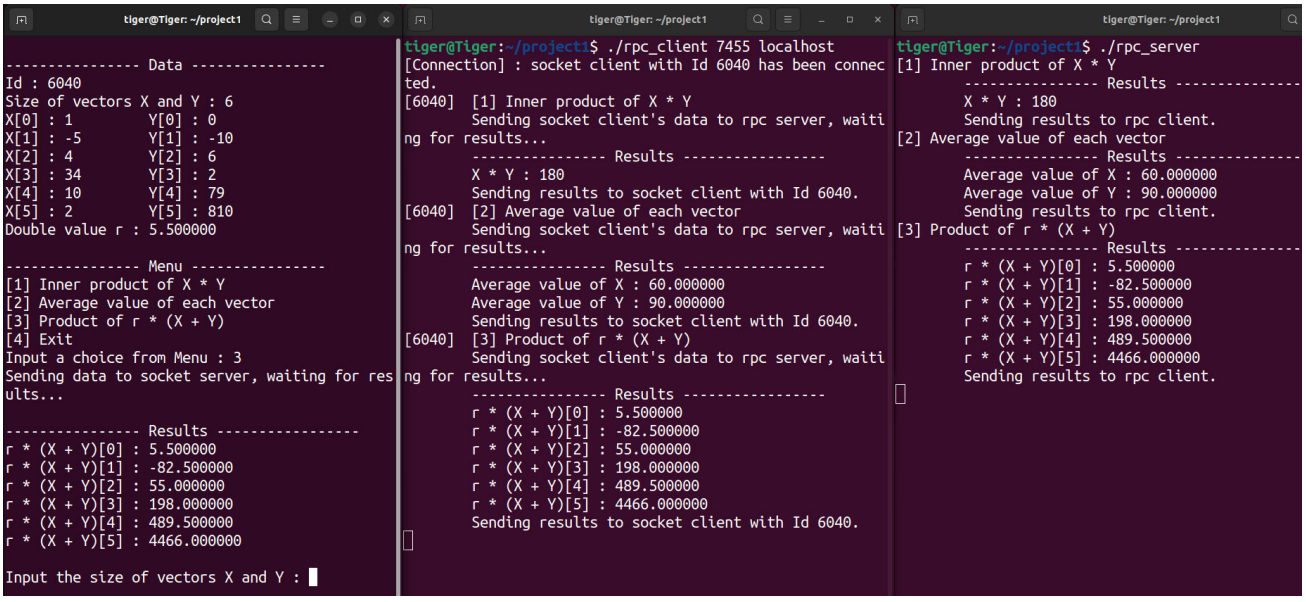
2nd Choice Results



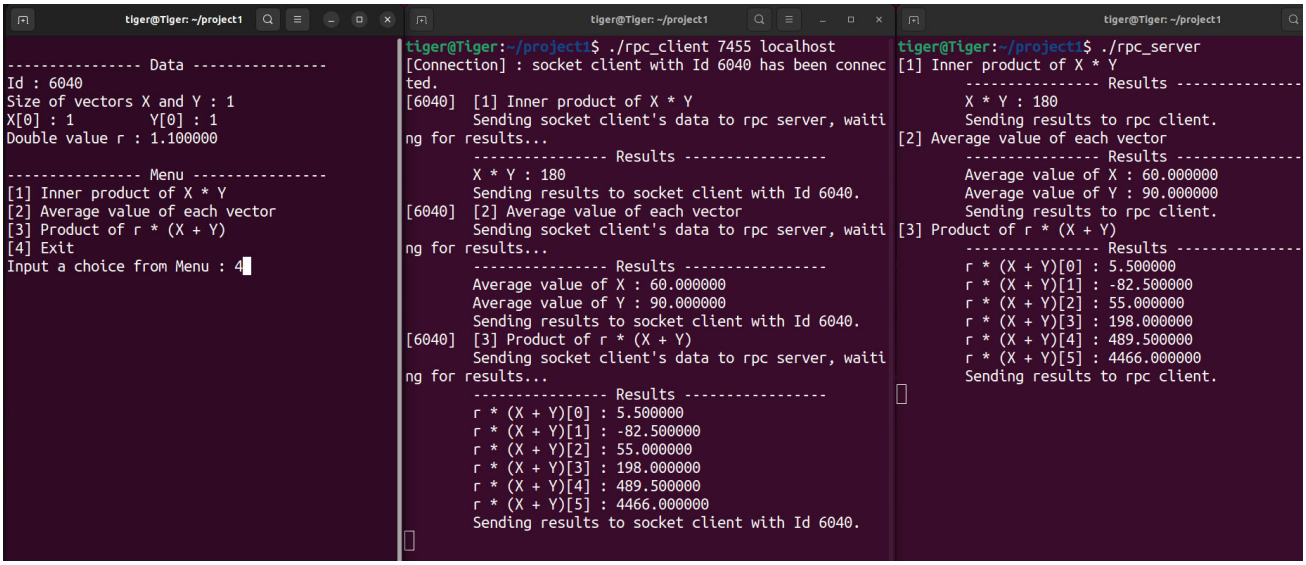
3rd Choice



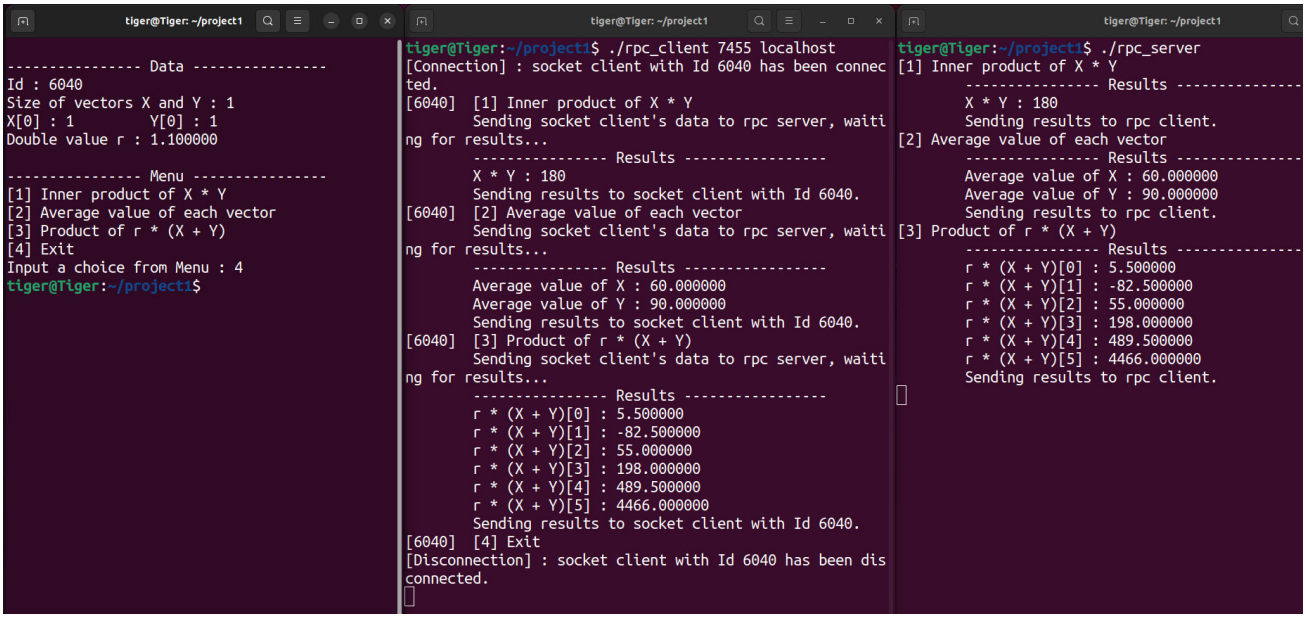
3rd Choice Results



4th Choice

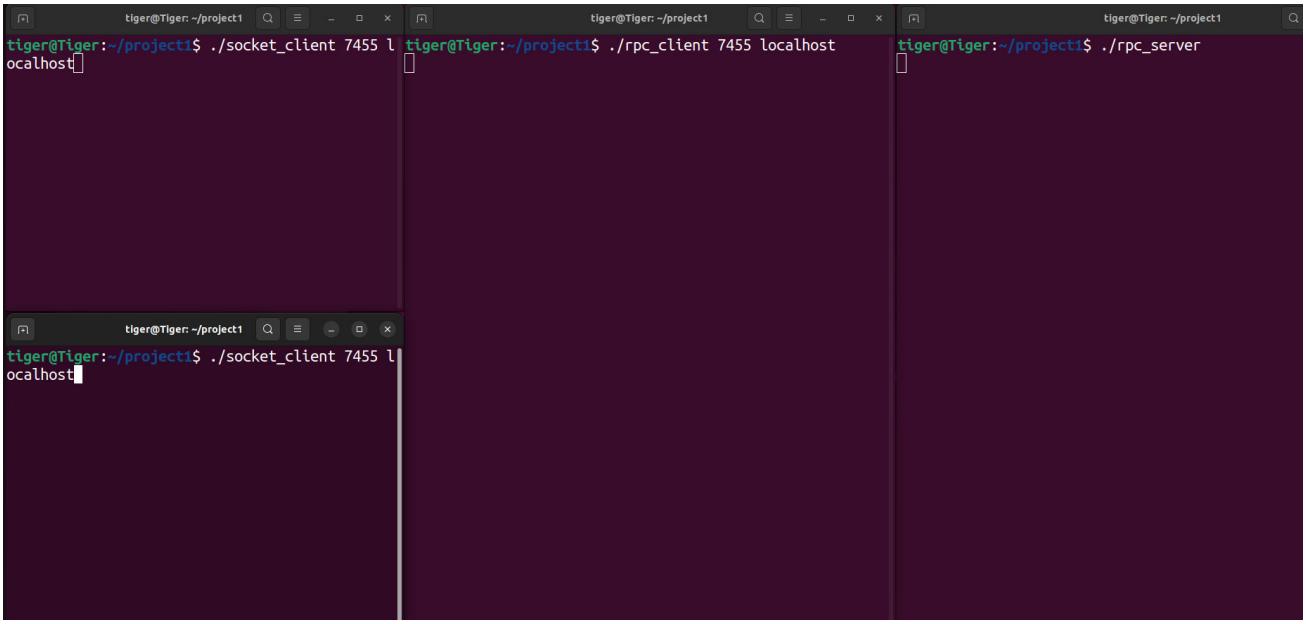


4th Choice Results

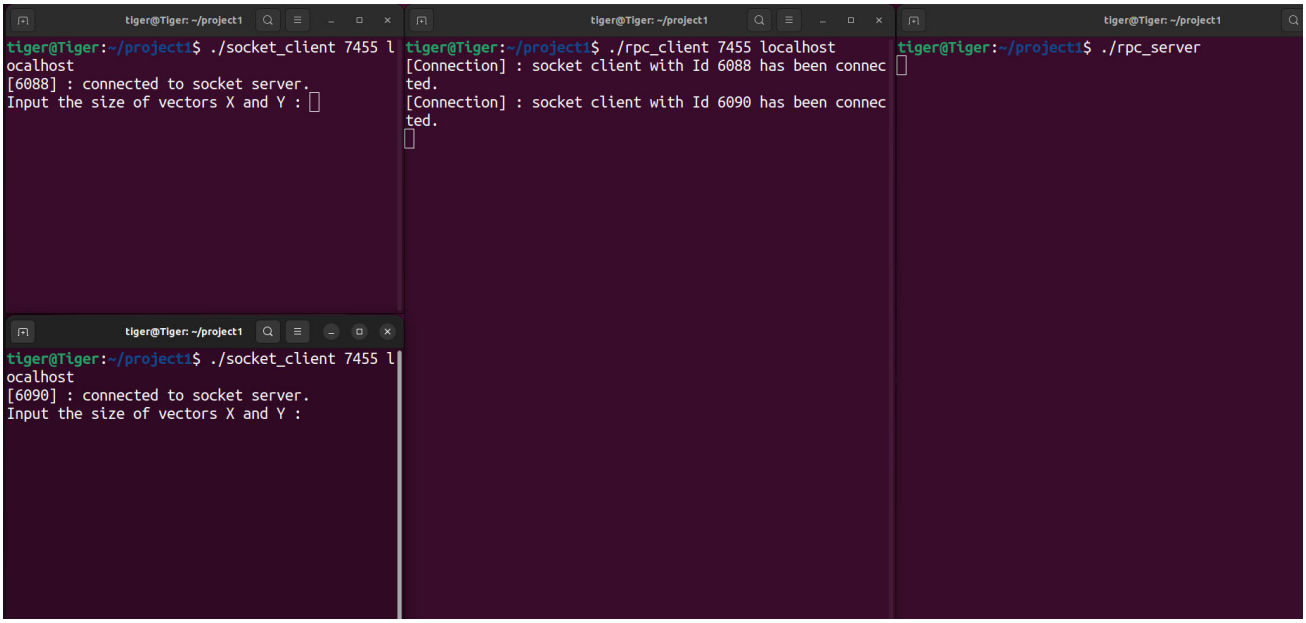


Socket clients: 2

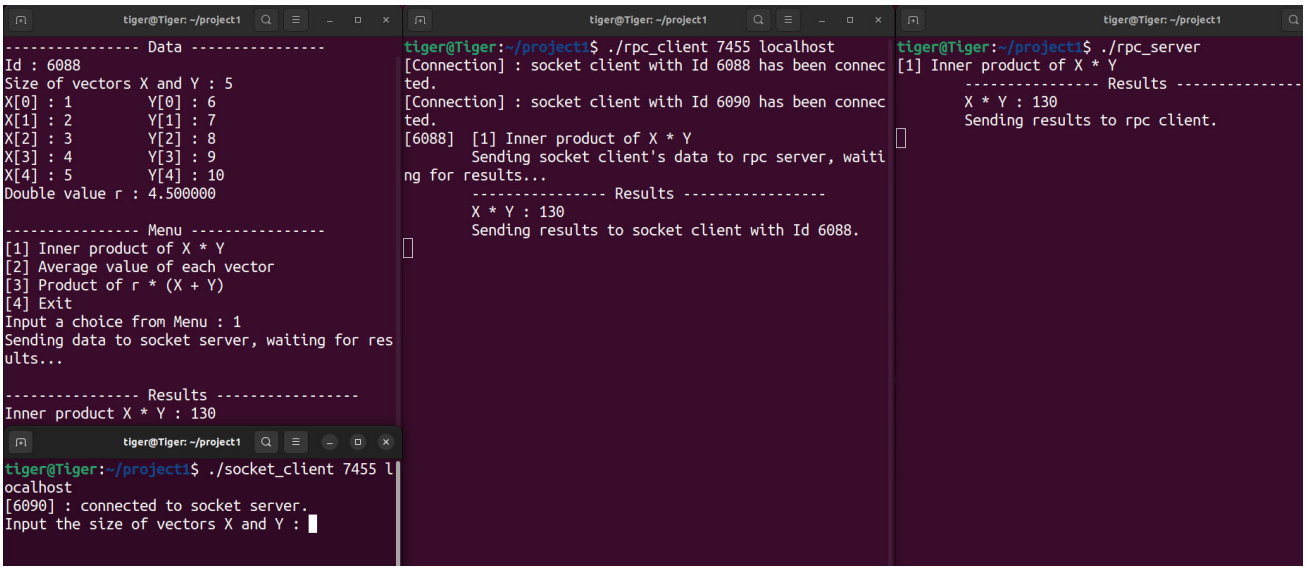
Servers On



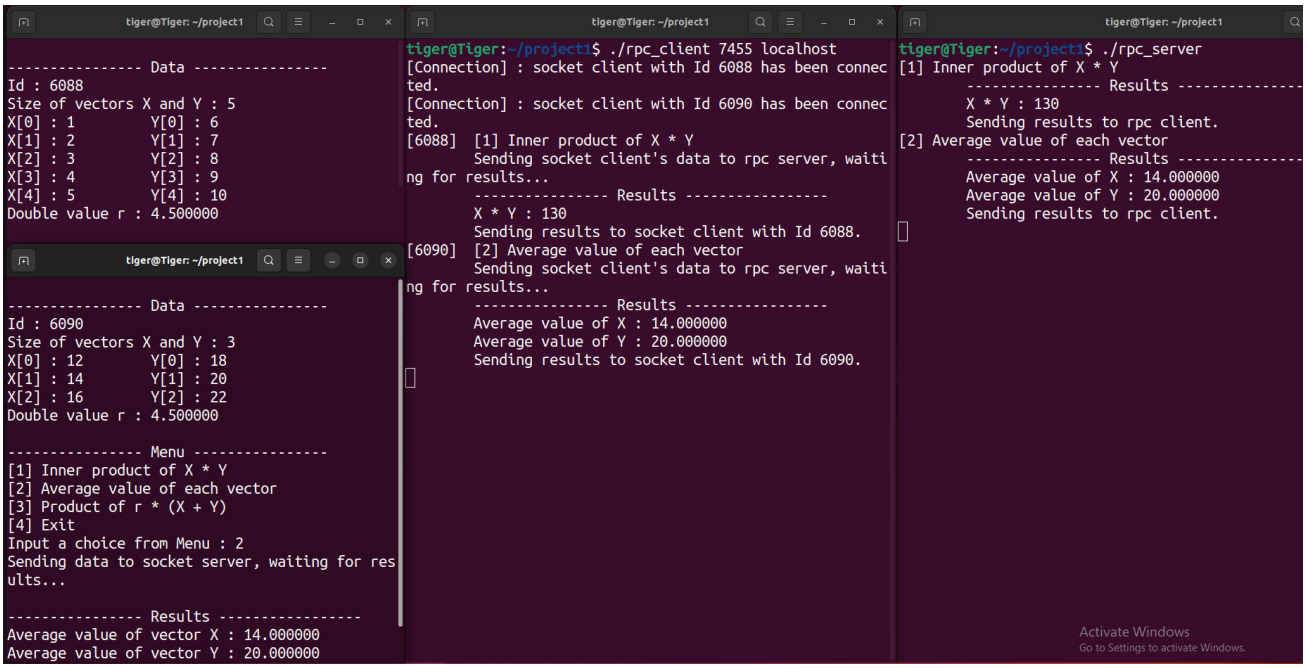
Connection



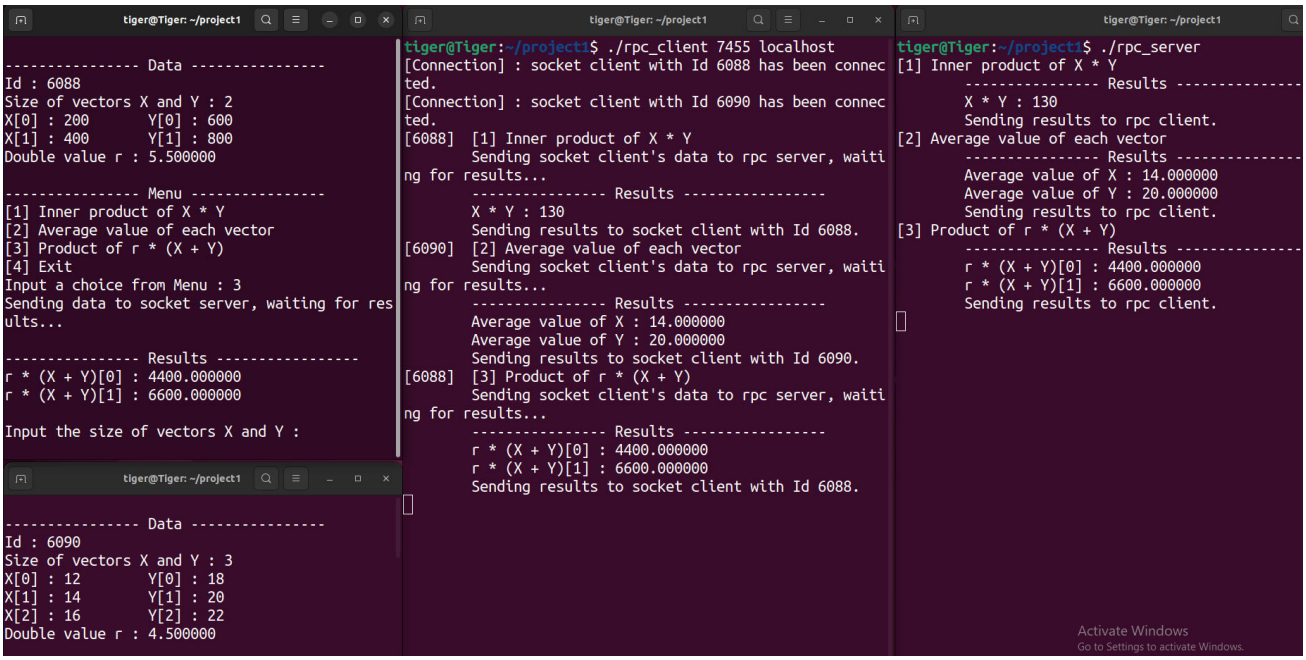
1st Choice Results



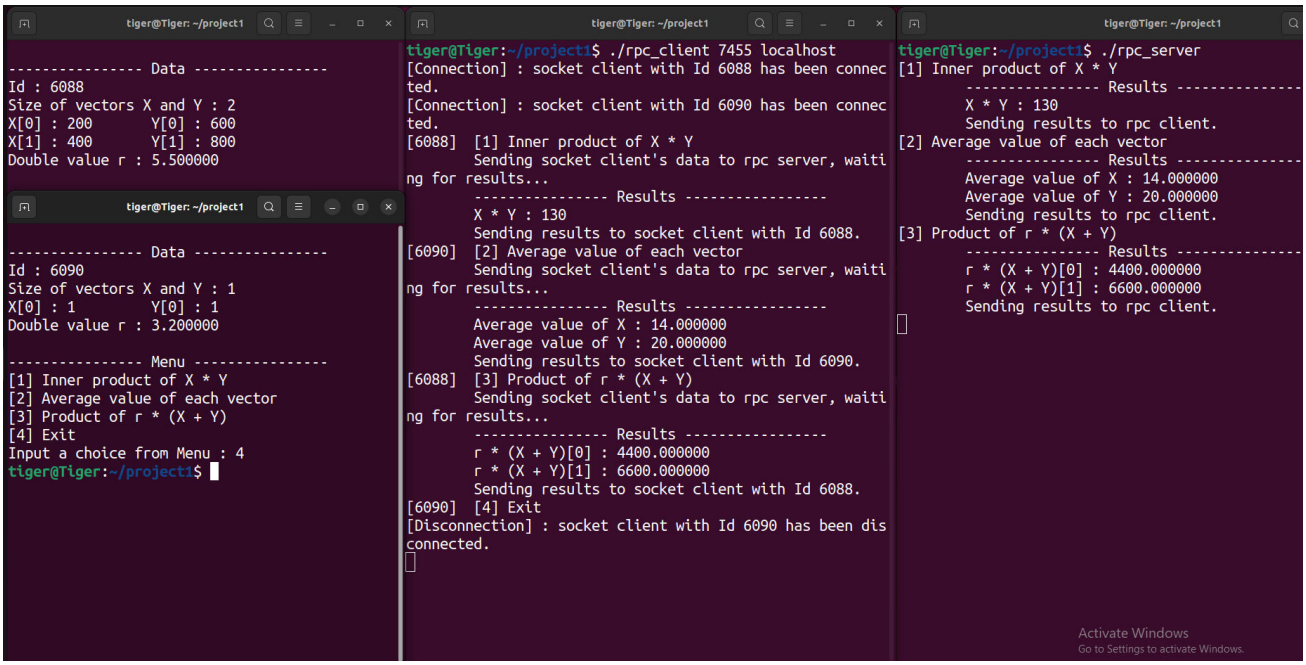
2nd Choice Results



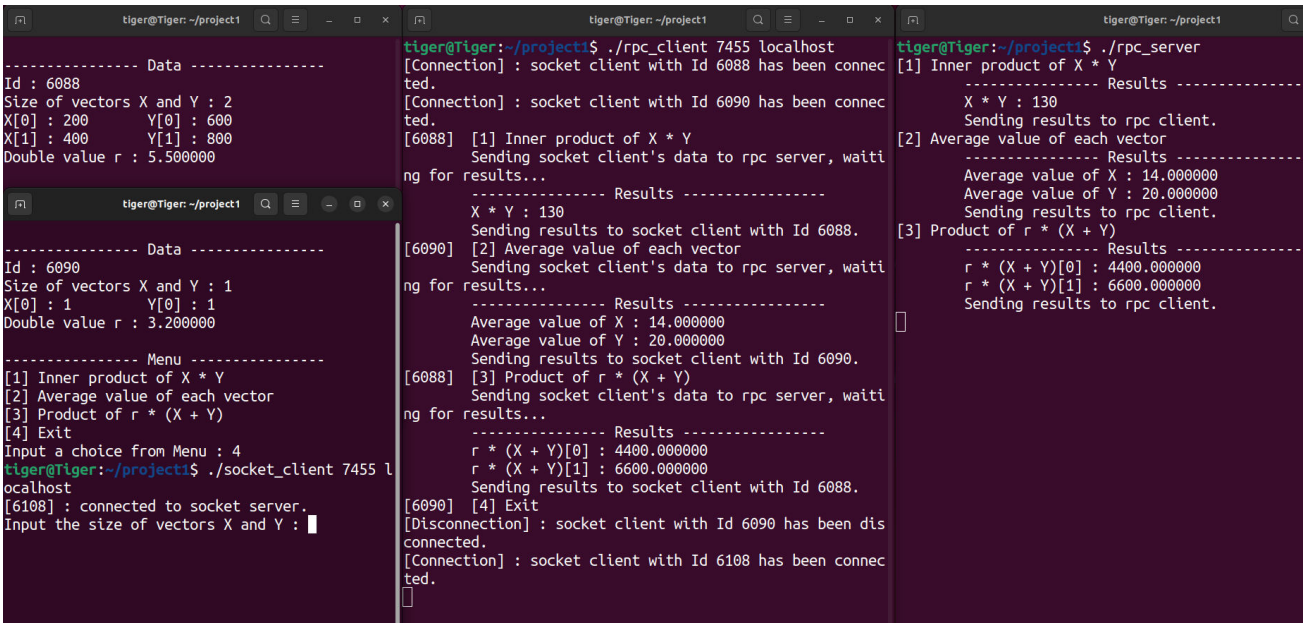
3rd Choice Results



4th Choice Results

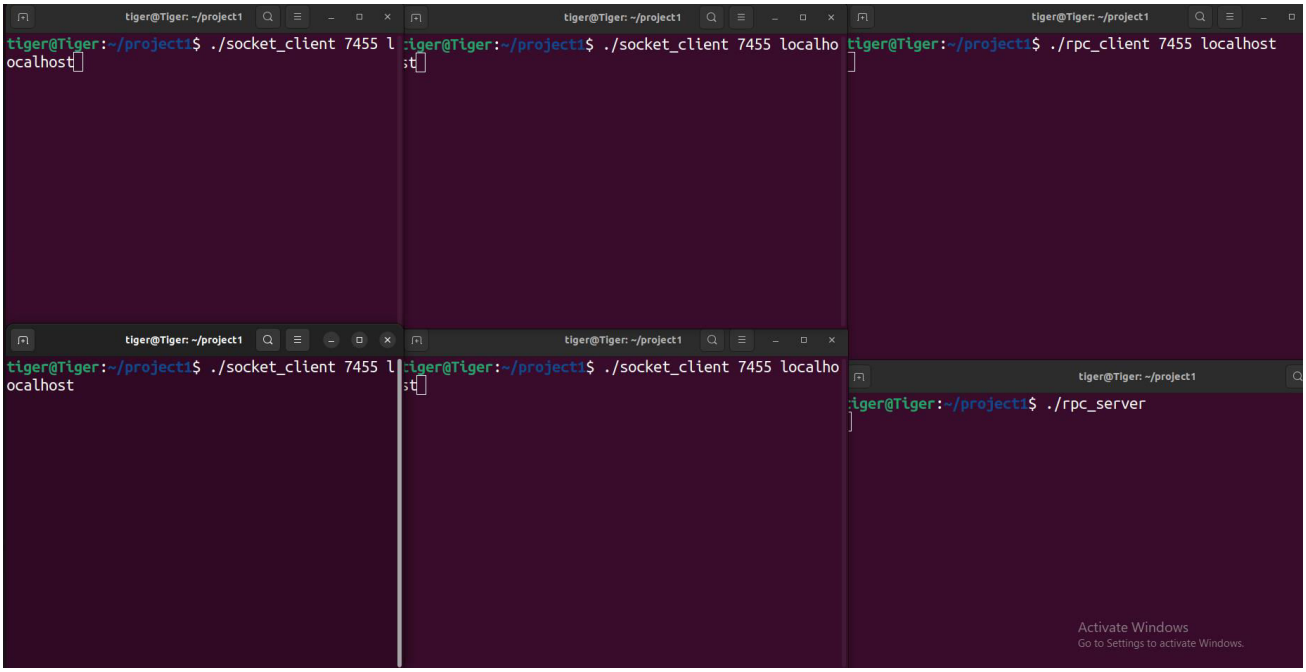


New Connection

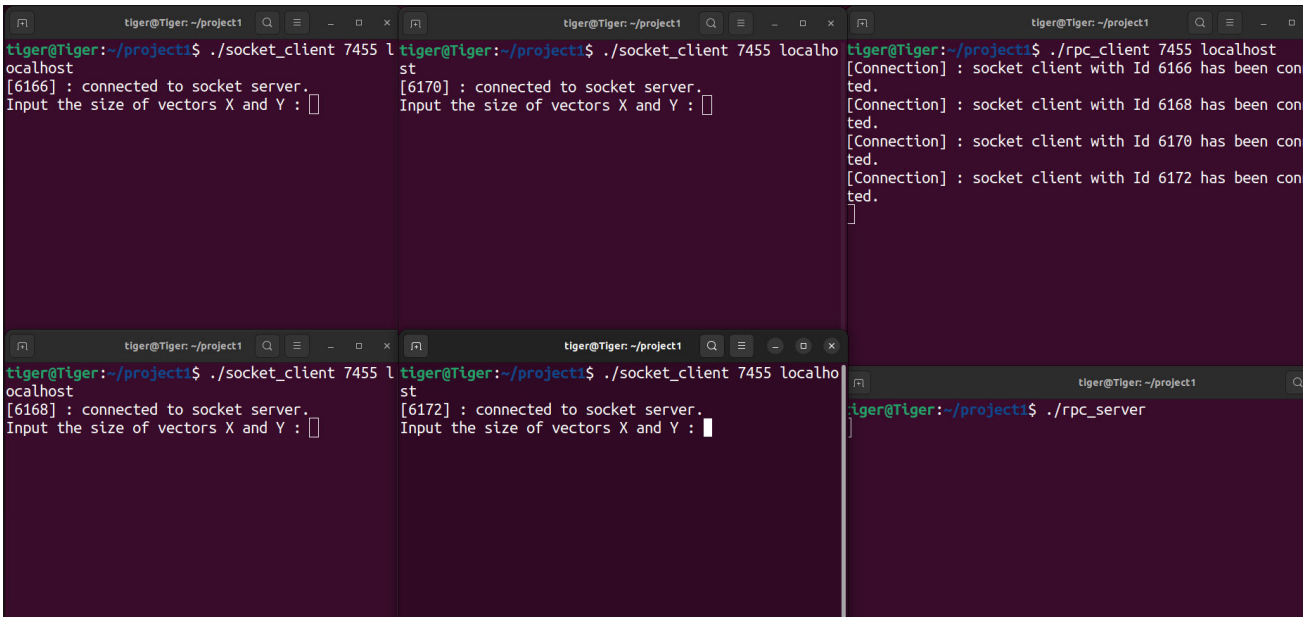


Socket clients: 4

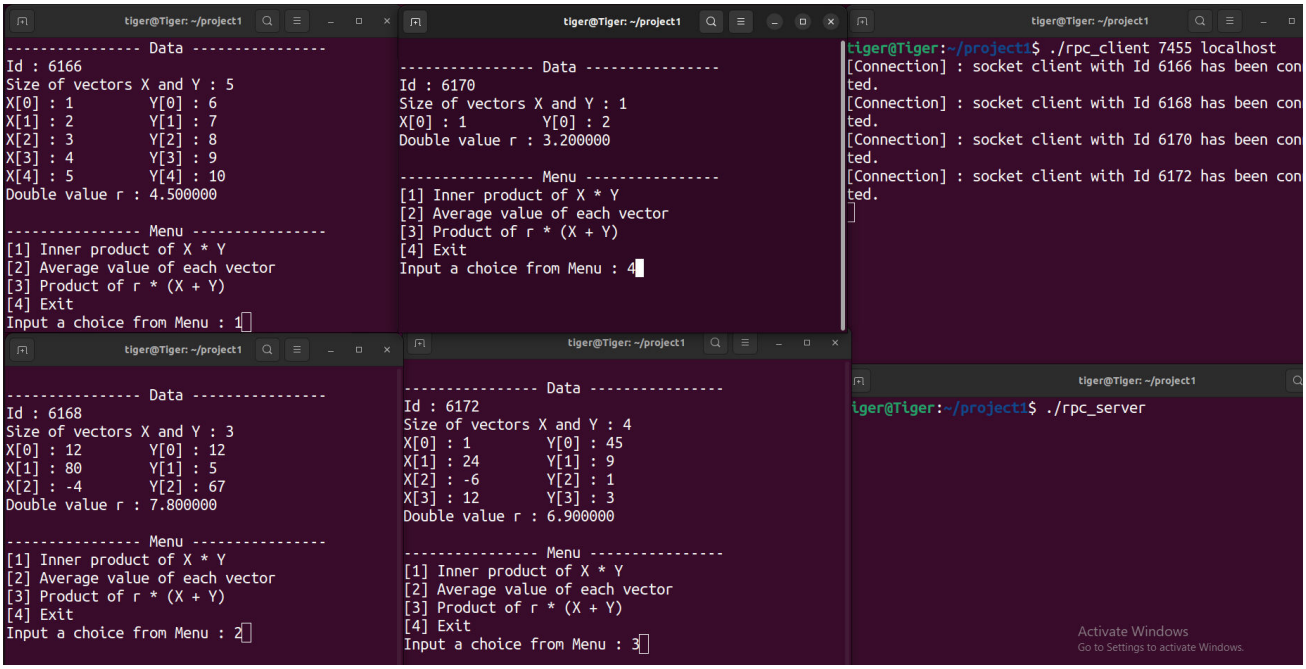
Servers On



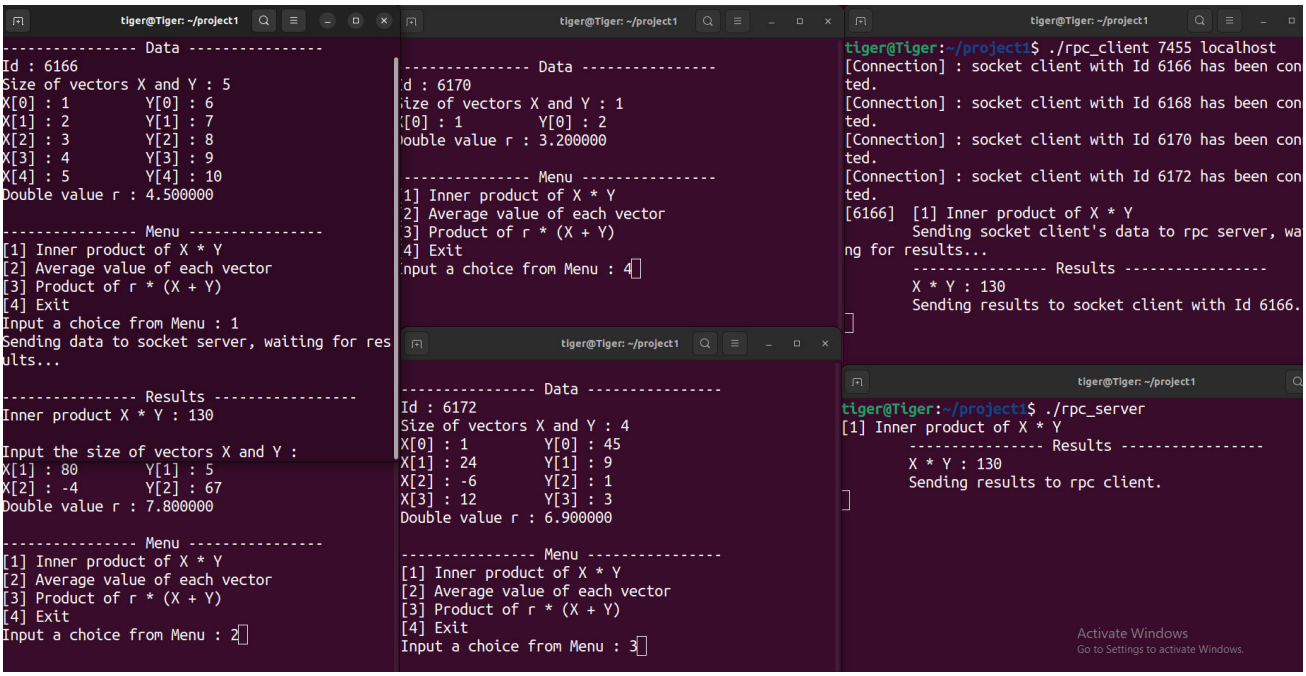
Connection



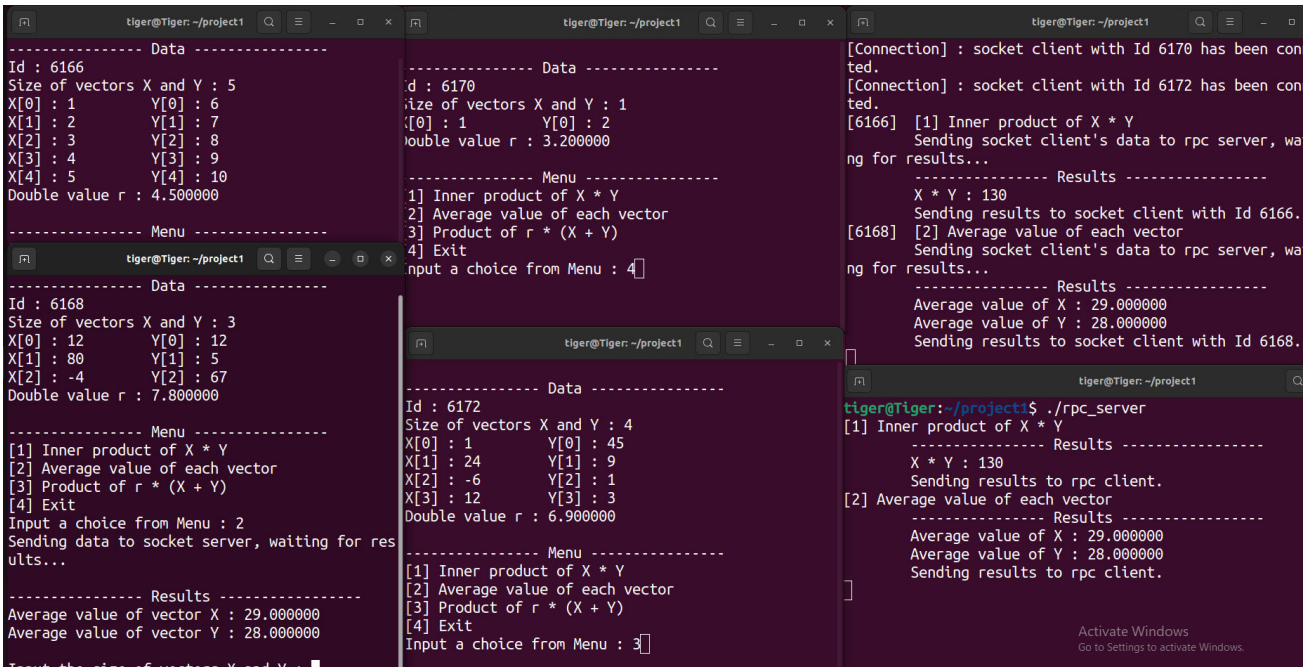
Input Data



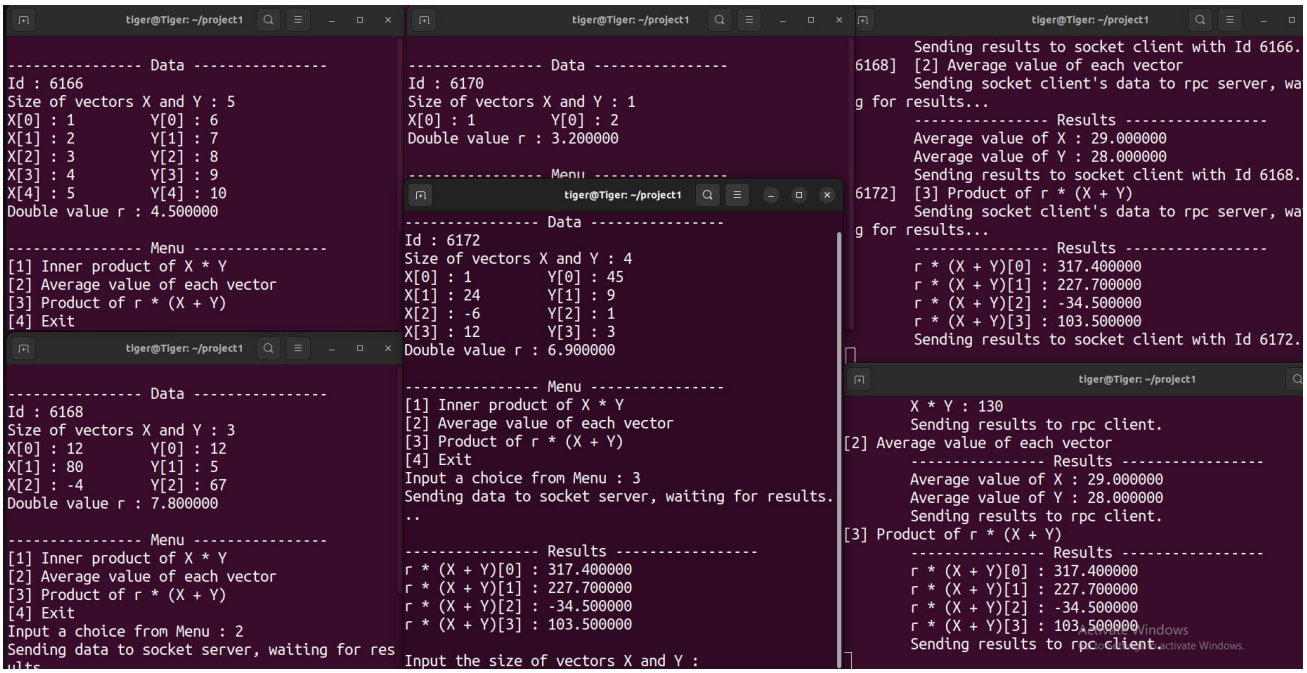
1st Choice Results



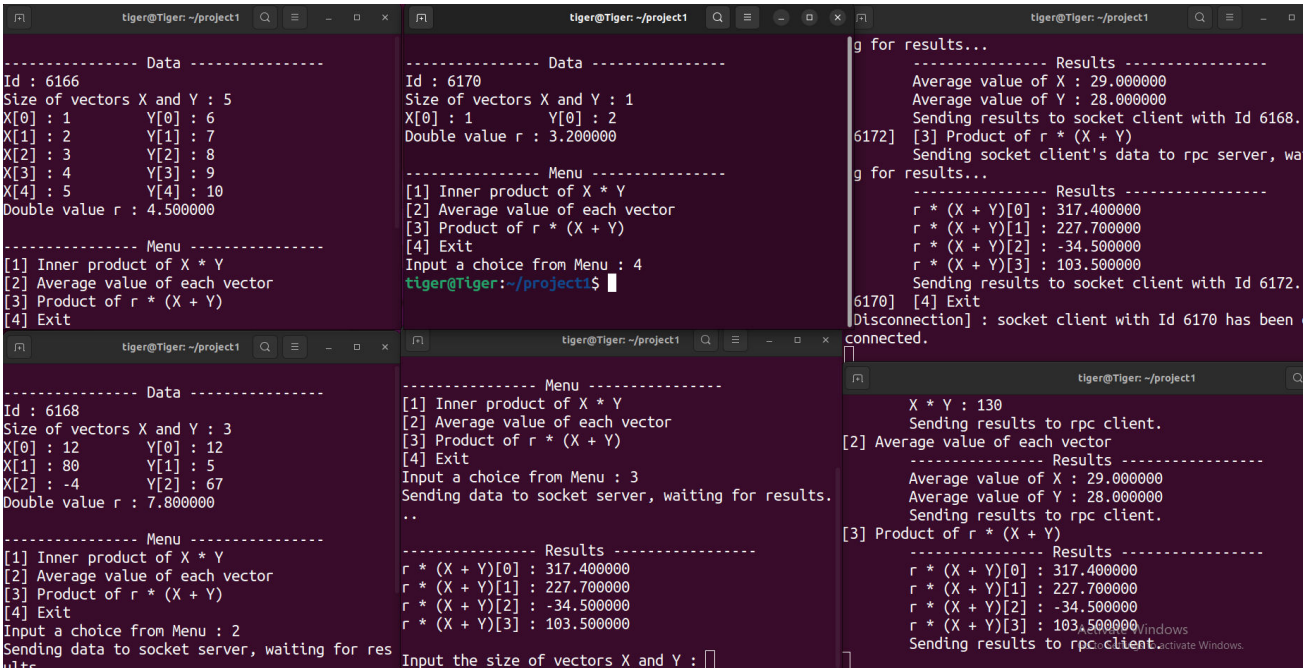
2nd Choice Results



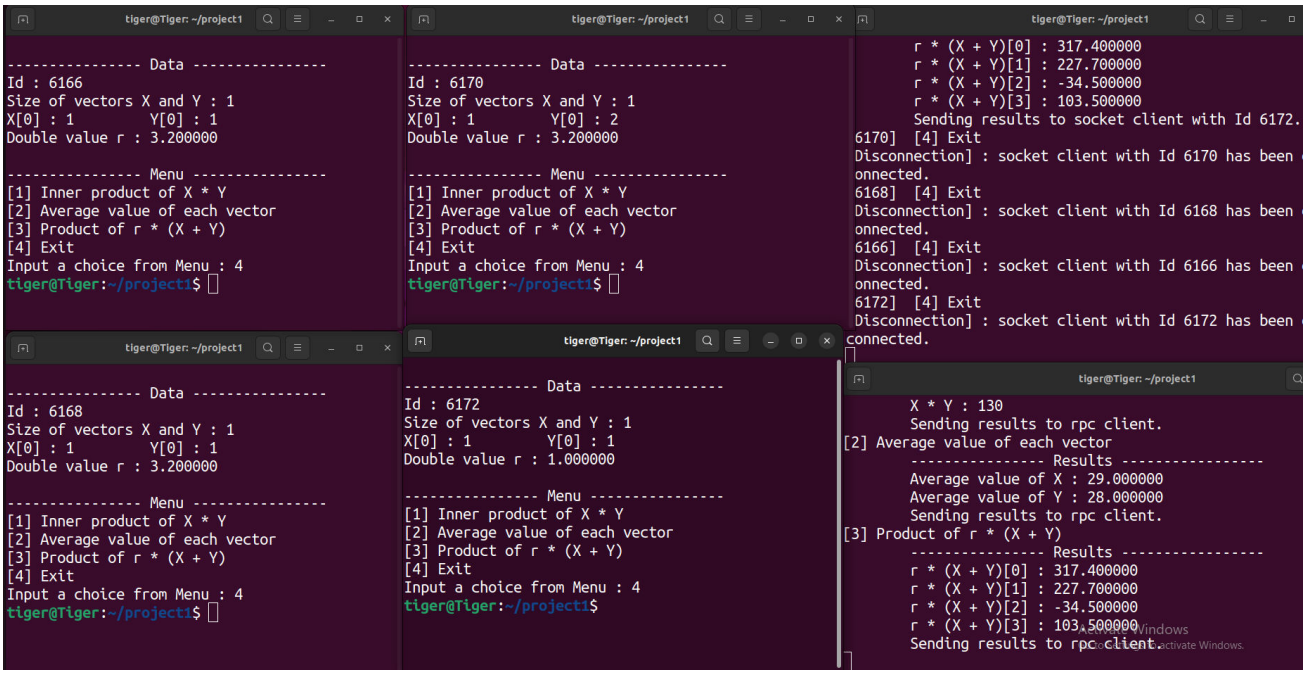
3rd Choice Results



4th Choice Results

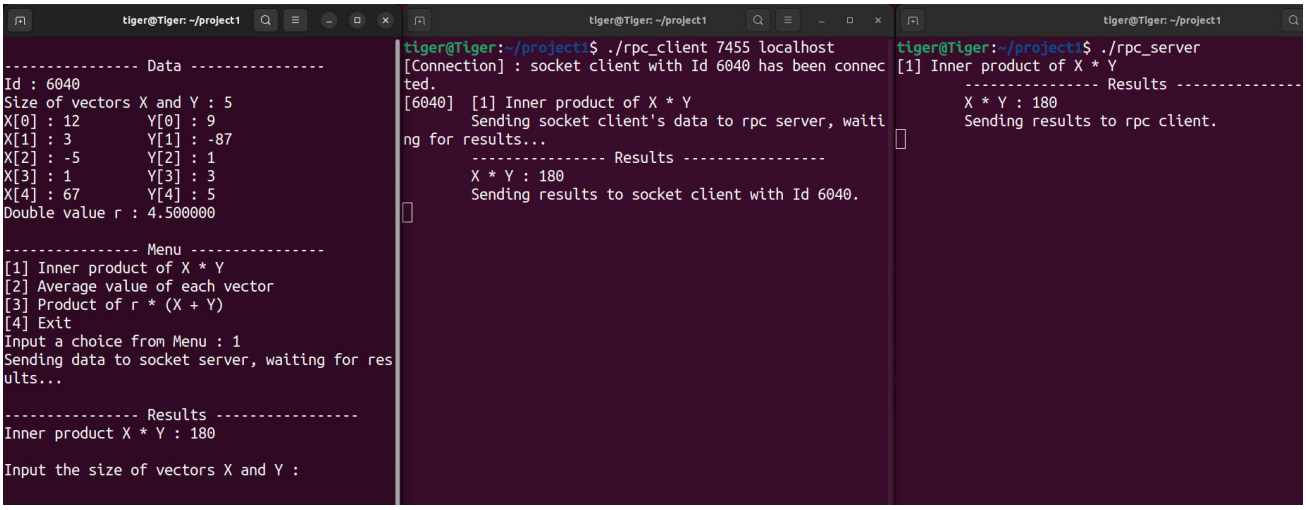


Disconnection



Verification of results

[1] Inner Product X * Y



vector	a
a_1	12
a_2	3
a_3	-5
a_4	1
a_5	67

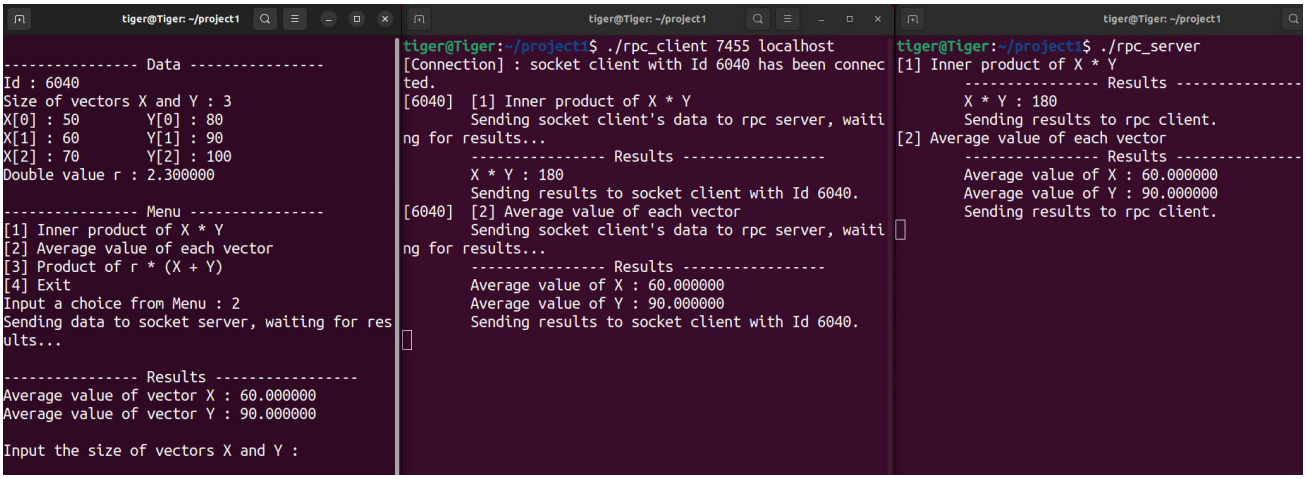
vector	b
b_1	9
b_2	-87
b_3	1
b_4	3
b_5	5

Execute | Clear | Store/Read | Print | 14digit

vector product

	product
inner product c	180
outer product c	no answer

[2] Mean value of the two vectors



Average Calculator

Please provide numbers separated by a comma to calculate the average of the numbers.

Result

Average: **60**

$$\text{Average} = \frac{\text{Sum}}{\text{Count}} = \frac{180}{3} = 60$$

Sum	180
Count	3
Median	60
Geometric Mean	59.439219527631
Largest	70
Smallest	50
Range	20

50, 60, 70

Calculate Clear

Average Calculator

Please provide numbers separated by a comma to calculate the average of the numbers.

Result

Average: **90**

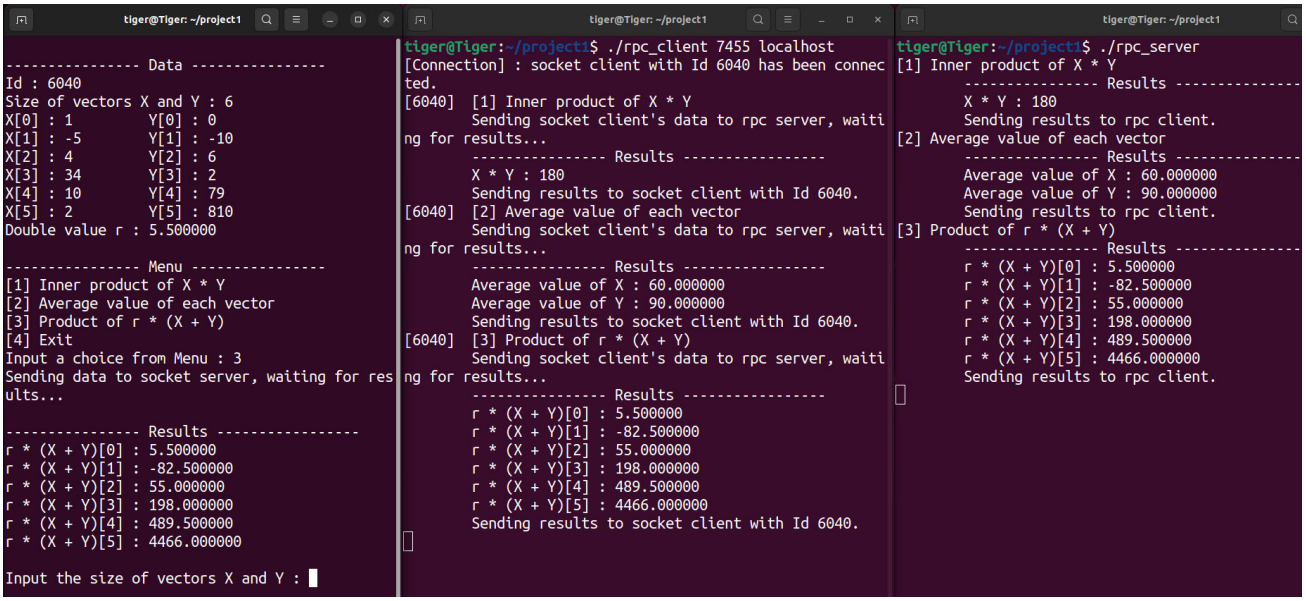
$$\text{Average} = \frac{\text{Sum}}{\text{Count}} = \frac{270}{3} = 90$$

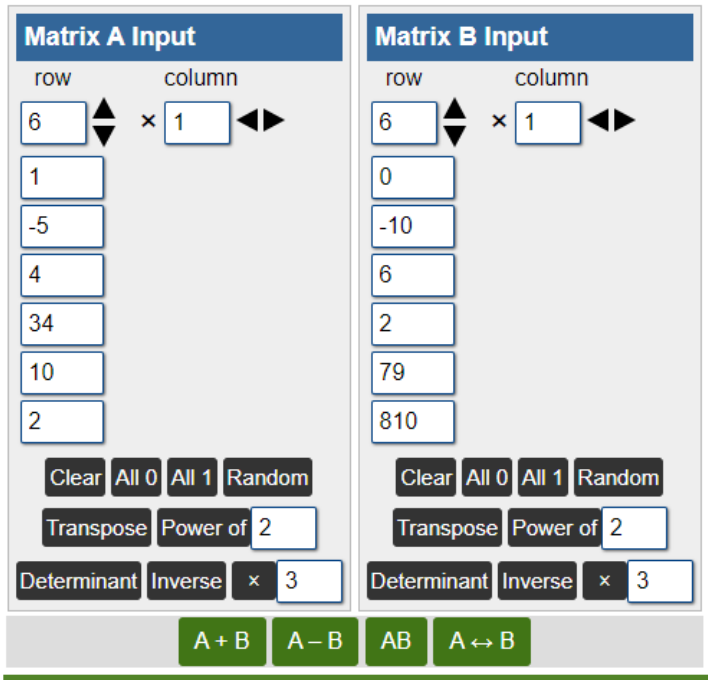
Sum	270
Count	3
Median	90
Geometric Mean	89.628094931143
Largest	100
Smallest	80
Range	20

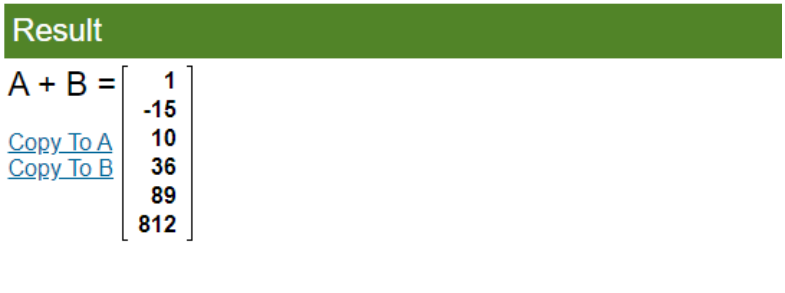
80, 90, 100

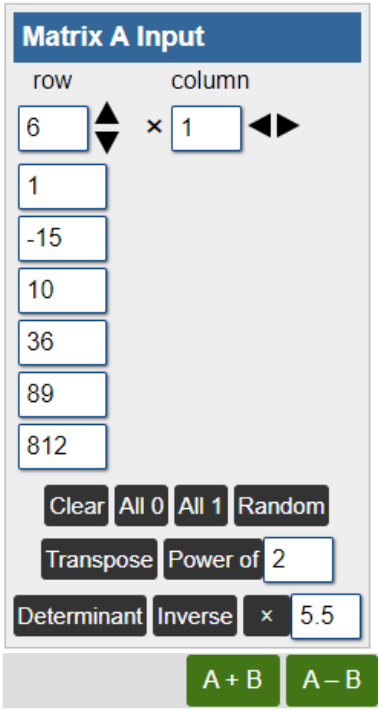
Calculate Clear

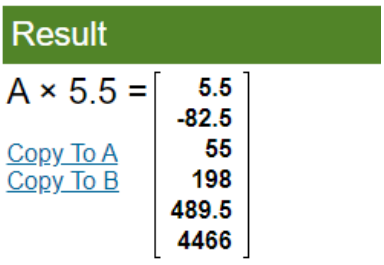
[3] Product r * (X + Y)











Thank you for your attention.