PROGRAM CODE

mta.c

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <string.h>

int resolveClient(char* mail_id) {
	FILE* fp = fopen("dns", "r");

	if(fp == NULL)
		return -1;

	char str[120];
	int index = -1;

	while(fgets(str, 120, fp) != NULL) {
		if(strstr(str, mail_id) == str && str[strlen(mail_id)] == ':') {
			index = ftell(fp) - strlen(str);
		}
	}

	if(index == -1)
		return -1;
	else {
		fseek(fp, index, SEEK_SET);
		fgets(str, 120, fp);
		str[strlen(str) - 1] = '\0';

		return atoi(str + strlen(mail_id) + 1);
	}
}

void serverRecv(int client_fd) {
	char str[100];
	char mail[1000];
	int mda_port;

	while(1) {
		int k = recv(client_fd, str, 100 * sizeof(char), 0);

		if(k < 0) {
			printf("Receive failed!\n");
			break;
		} else if(k == 0 || !strcmp(str, "QUIT")) {
			printf("Connection closed.\n");
			break;
		} else if(!strcmp(str, "HELO")) {
			mail[0] = '\0';
			mda_port = -1;
		} else if(strstr(str, "MAIL FROM ") == str) {
			strcat(mail, str);
			strcat(mail, "\n");

			printf("Sender: %s\n", str + 10);

			str[0] = '\0';
			strcat(str, "200 OK");
```

```
                if(send(client_fd, str, (strlen(str) + 1) * sizeof(char), 0) <
0) {
                        printf("Send failed!\n");
                        break;
                }
        } else if(strstr(str, "RCP TO ") == str) {
                strcat(mail, str);
                strcat(mail, "\n");

                printf("Receiver: %s\n", str + 7);

                if((mda_port = resolveClient(str + 7)) == -1) {
                        str[0] = '\0';
                        strcat(str, "400 BAD");
                } else {
                        str[0] = '\0';
                        strcat(str, "200 OK");
                }

                if(send(client_fd, str, (strlen(str) + 1) * sizeof(char), 0) <
0) {
                        printf("Send failed!\n");
                        break;
                }
        } else if(strstr(str, "DATA ") == str) {
                strcat(mail, "DATA\n");
                strcat(mail, str + 5);
                strcat(mail, "\n");

                printf("Data:\n%s\n", str + 5);

                str[0] = '\0';
                strcat(str, "200 OK");

                if(send(client_fd, str, (strlen(str) + 1) * sizeof(char), 0) <
0) {
                        printf("Send failed!\n");
                        break;
                }

                struct sockaddr_in address;

                address.sin_family = AF_INET;
                address.sin_addr.s_addr = INADDR_ANY;
                address.sin_port = htons(mda_port);

                close(client_fd);

                if((client_fd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
                        printf("Socket creation failed!\n");
                        break;
                }

                if(connect(client_fd, (struct sockaddr*) &address,
sizeof(address)) < 0) {
                        printf("Connection failed!\n");
                        break;
                }

                if(send(client_fd, mail, (strlen(mail) + 1) * sizeof(char), 0)
< 0) {
                        printf("Send failed!\n");
                        break;
```

```
                } else {
                        printf("Forwarded to MDA %d\n", mda_port);
                }
            } else {
                printf("Received invalid request from client: %s\n", str);
                break;
            }
        }

        close(client_fd);
}

void main() {
        int server_fd, client_fd, PORT;
        struct sockaddr_in address, cli_addr1, cli_addr2;
        int addrlen = sizeof(address);

        printf("SMTP MTA\n");

        printf("Enter port: ");
        scanf("%d", &PORT);

        if((server_fd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
            printf("Socket creation failed!\n");
            exit(1);
        } else {
            printf("Server socket created.\n");
        }

        address.sin_family = AF_INET;
        address.sin_addr.s_addr = INADDR_ANY;
        address.sin_port = htons(PORT);

        if(bind(server_fd, (struct sockaddr*) &address, addrlen) < 0) {
            printf("Socket binding failed!\n");
            exit(1);
        }

        if(listen(server_fd, 5) < 0) {
            printf("Listening failed!\n");
            exit(1);
        }

        while(1) {
            if((client_fd = accept(server_fd, (struct sockaddr*) &address,
(socklen_t*) &addrlen)) < 0) {
                printf("Connection failed!\n");
                return;
            }

            printf("Connected to client.\n");

            serverRecv(client_fd);
        }

        close(server_fd);
}
```

mda.c

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/socket.h>
```

```
#include <arpa/inet.h>
#include <unistd.h>
#include <string.h>

int m_len = 0;
char mails[50][1000];

void serverRecv(int client_fd) {
      char str[1000];

      while(1) {
            int k = recv(client_fd, str, 1000 * sizeof(char), 0);

            if(k < 0) {
                  printf("Receive failed!\n");
                  return;
            } else if(k == 0) {
                  printf("Connection closed.\n");
                  return;
            } else if(strstr(str, "MAIL FROM ") == str) {
                  strcpy(mails[m_len], str);
                  m_len++;

                  printf("Mail received and queued!\n");
                  break;
            } else if(strstr(str, "RECV MAIL ") == str) {
                  char* mail_id = strdup(str + 10);

                  for(int i = 0; i < m_len; i++) {
                        char* recp = strstr(mails[i], "RCP TO ");
                        recp = strdup(recp + 7);

                        recp = strtok(recp, "\n");

                        if(!strcmp(mail_id, recp)) {
                              if(send(client_fd, mails[i], (strlen(mails[i]) +
1) * sizeof(char), 0) < 0) {
                                    printf("Send failed!\n");
                                    break;
                              }
                        } else {
                              printf("Recp: %s", recp);
                        }
                  }

                  char* str1 = "200 OK";

                  if(send(client_fd, str1, (strlen(str1) + 1) * sizeof(char), 0)
< 0) {
                        printf("Send failed!\n");
                        break;
                  } else {
                        printf("Queued mails sent to %s.\n", mail_id);
                        break;
                  }
            } else {
                  printf("Invalid request!\n");
                  break;
            }
      }

      close(client_fd);
}
```

```
void main() {
	int server_fd, client_fd, PORT;
	struct sockaddr_in address, cli_addr1, cli_addr2;
	int addrlen = sizeof(address);

	printf("SMTP MDA\n");

	printf("Enter port: ");
	scanf("%d", &PORT);

	if((server_fd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
		printf("Socket creation failed!\n");
		exit(1);
	} else {
		printf("Server socket created.\n");
	}

	address.sin_family = AF_INET;
	address.sin_addr.s_addr = INADDR_ANY;
	address.sin_port = htons(PORT);

	if(bind(server_fd, (struct sockaddr*) &address, addrlen) < 0) {
		printf("Socket binding failed!\n");
		exit(1);
	}

	if(listen(server_fd, 5) < 0) {
		printf("Listening failed!\n");
		exit(1);
	}

	while(1) {
		if((client_fd = accept(server_fd, (struct sockaddr*) &address,
(socklen_t*) &addrlen)) < 0) {
			printf("Connection failed!\n");
			continue;
		}

		printf("Connected to client.\n");

		serverRecv(client_fd);
	}

	close(server_fd);
}
```

mua.c

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <string.h>

void closeConn(int client_fd) {
	char* str = "QUIT";
	printf("Connection closed.\n");
	send(client_fd, str, (strlen(str) + 1) * sizeof(char), 0);
	close(client_fd);
}

void clientSend(struct sockaddr_in serv_addr, char* mail_id) {
```

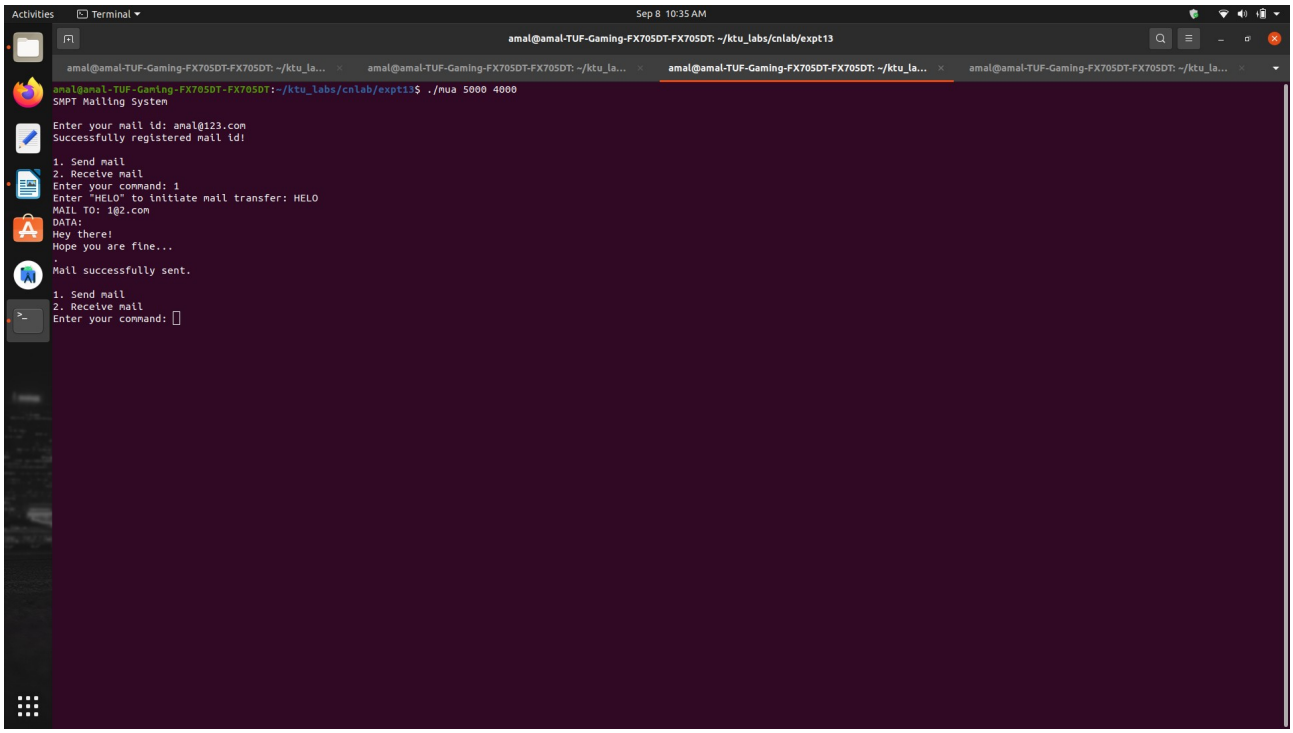
```
	char str[100];
	char str1[1000];
	int client_fd;

	printf("Enter \"HELO\" to initiate mail transfer: ");
	scanf("%s", str);
	getchar();

	if(!strcmp(str, "HELO")) {
		if((client_fd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
			printf("Socket creation failed!\n");
			return;
		}

		if(connect(client_fd, (struct sockaddr*) &serv_addr,
sizeof(serv_addr)) < 0) {
			printf("Connection failed!\n");
			close(client_fd);
			return;
		}

		if(send(client_fd, str, 100 * sizeof(char), 0) < 0) {
			printf("Send failed!\n");
			closeConn(client_fd);
			return;
		}
	} else {
		printf("Invalid input!\n");
		return;
	}

	if(strlen(str) != 0) {
		str1[0] = '\0';
		strcat(str1, "MAIL FROM ");
		strcat(str1, mail_id);

		if(send(client_fd, str1, (strlen(str1) + 1) * sizeof(char), 0) < 0)
{
			printf("Send failed!\n");
			closeConn(client_fd);
			return;
		} else {
			if (recv(client_fd, str, 100 * sizeof(char), 0) < 0) {
				printf("Receive failed!\n");
				closeConn(client_fd);
				return;
			} else {
				if(strcmp(str, "200 OK") != 0) {
					printf("Bad response!\n");
					closeConn(client_fd);
					return;
				}
			}
		}
	} else {
		printf("Invalid response!\n");
		closeConn(client_fd);
		return;
	}

	printf("MAIL TO: ");
	fgets(str, 100, stdin);
	str[strlen(str) - 1] = '\0';
```

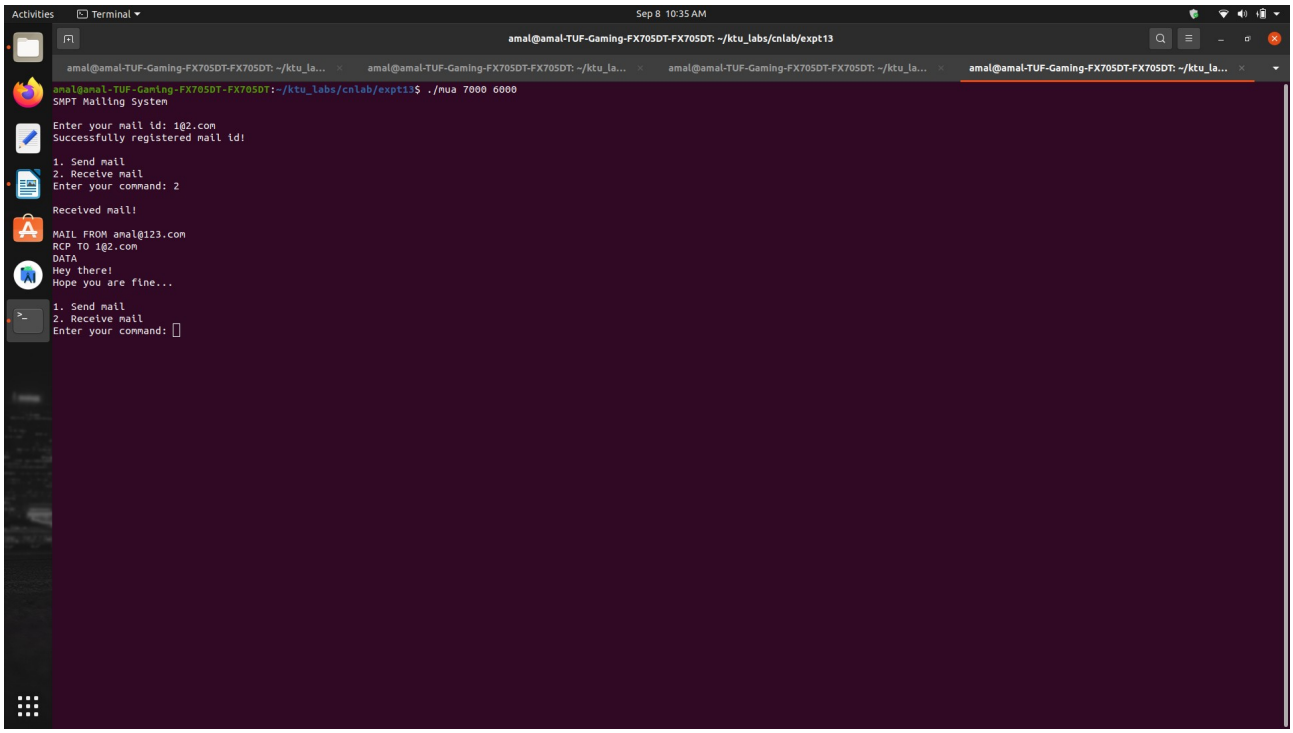
```
	if(strlen(str) != 0) {
		str1[0] = '\0';
		strcat(str1, "RCP TO ");
		strcat(str1, str);

		if(send(client_fd, str1, (strlen(str1) + 1) * sizeof(char), 0) < 0)
{
			printf("Send failed!\n");
			closeConn(client_fd);
			return;
		} else {
			if (recv(client_fd, str, 100 * sizeof(char), 0) < 0) {
				printf("Receive failed!\n");
				closeConn(client_fd);
				return;
			} else {
				if(strcmp(str, "200 OK") != 0) {
					printf("Bad response!\n");
					closeConn(client_fd);
					return;
				}
			}
		}
	} else {
		printf("Invalid response!\n");
		closeConn(client_fd);
		return;
	}

	str1[0] = '\0';
	strcat(str1, "DATA ");
	printf("DATA:\n");

	while(1) {
		fgets(str, 100, stdin);

		if(!strcmp(str, ".\n")) {
			str1[strlen(str1) - 1] = '\0';

			if(send(client_fd, str1, (strlen(str1) + 1) * sizeof(char), 0)
< 0) {
				printf("Send failed!\n");
				closeConn(client_fd);
				return;
			} else {
				if (recv(client_fd, str, 100 * sizeof(char), 0) < 0) {
					printf("Receive failed!\n");
					closeConn(client_fd);
					return;
				} else {
					if(!strcmp(str, "200 OK")) {
						printf("Mail successfully sent.\n");
						return;
					} else {
						printf("Bad response!\n");
						closeConn(client_fd);
						return;
					}
				}
			}

			break;
		} else {
			strcat(str1, str);
```

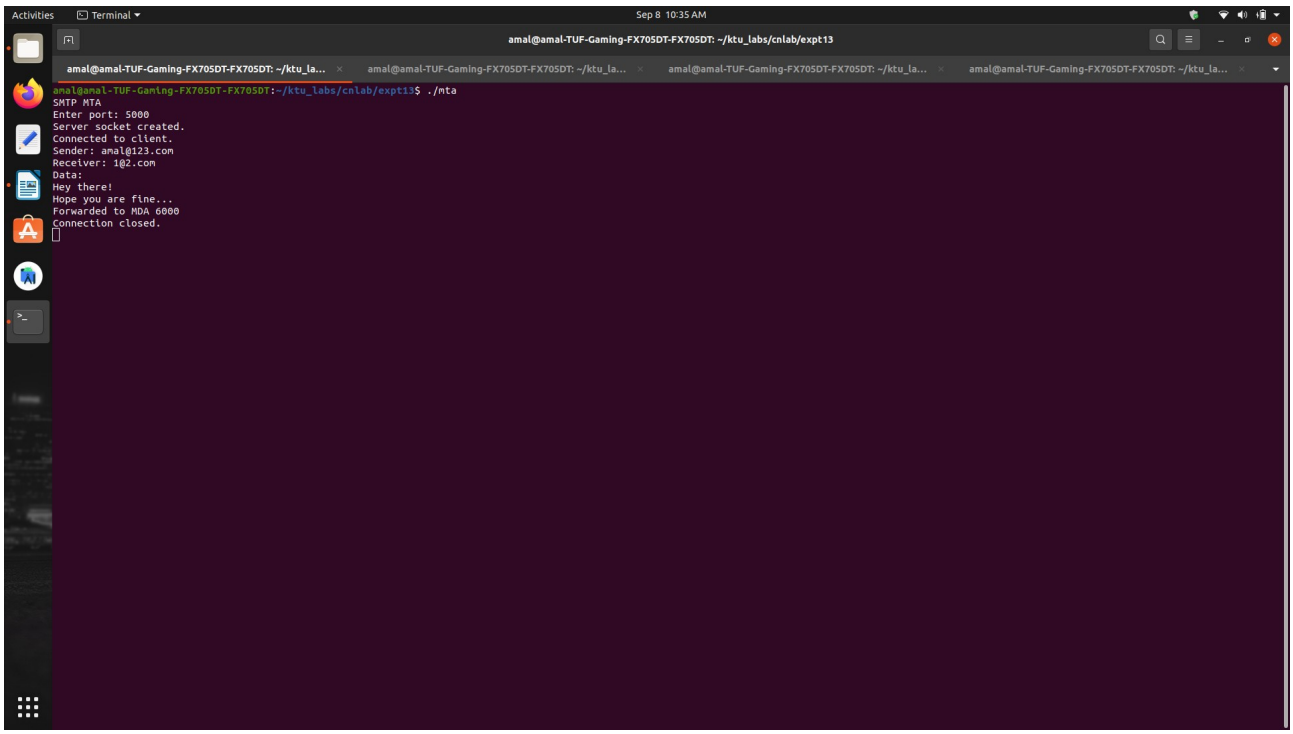
```
            }
      }

      closeConn(client_fd);
}

void clientRecv(struct sockaddr_in serv_addr, char* mail_id) {
      char str[1000];
      int client_fd;

      if((client_fd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
            printf("Socket creation failed!\n");
            return;
      }

      if(connect(client_fd,(struct sockaddr*) &serv_addr, sizeof(serv_addr)) <
0) {
            printf("Connection failed!\n");
            close(client_fd);
            return;
      }

      strcpy(str, "RECV MAIL ");
      strcat(str, mail_id);

      if(send(client_fd, str, 1000 * sizeof(char), 0) < 0) {
            printf("Send failed!\n");
            return;
      }

      while(1) {
            int k = recv(client_fd, str, 1000 * sizeof(char), 0);

            if(!strcmp(str, "200 OK"))
                  break;
            else if(k < 0) {
                  printf("Receive failed!\n");
                  break;
            } else {
                  printf("\nReceived mail!\n\n%s", str);
            }
      }

      close(client_fd);
}

void registerClient(char* str, int port) {
      FILE* fp = fopen("dns", "a");

      strcat(str, ":");

      char portStr[10];
      sprintf(portStr, "%d", port);

      strcat(str, portStr);
      strcat(str, "\n");

      fputs(str, fp);

      printf("Successfully registered mail id!\n");

      fclose(fp);
}
```

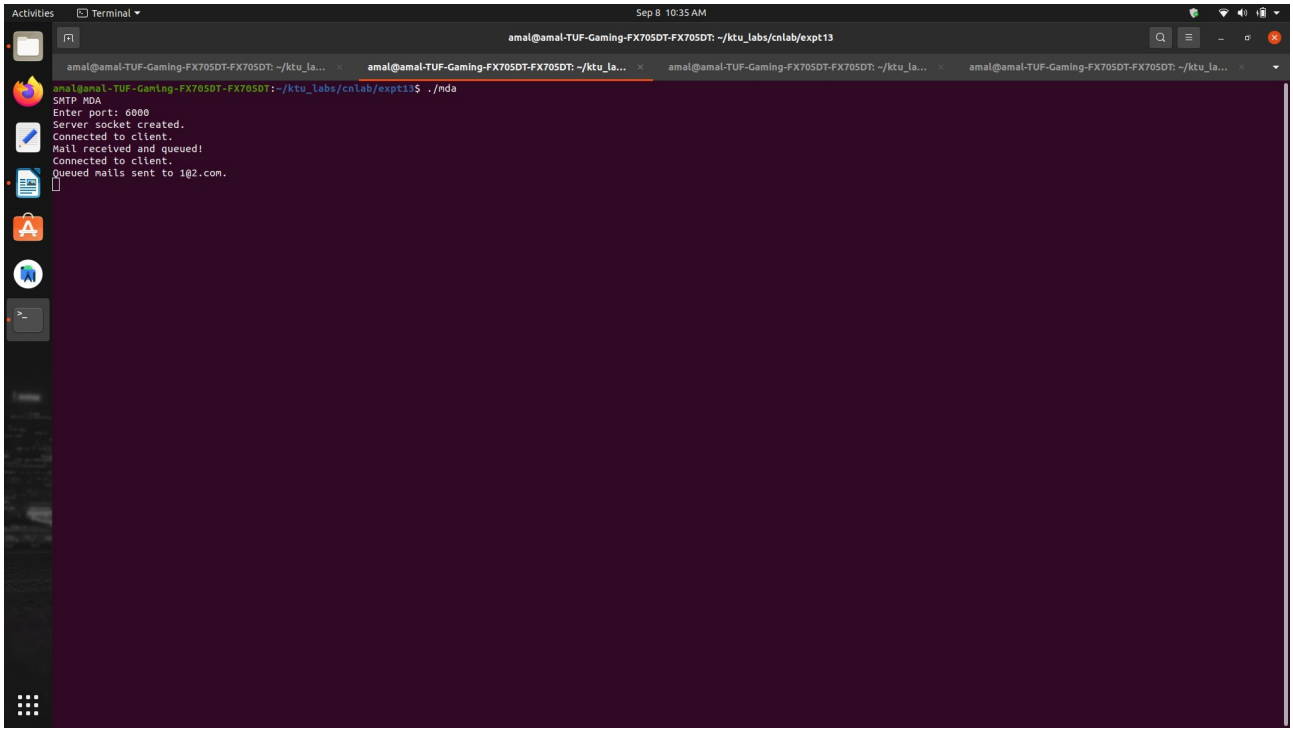
```
void main(int argc, char *argv[]) {
	int port1, port2;
	struct sockaddr_in serv_addr;

	if(argc == 3) {
		port1 = atoi(argv[1]);
		port2 = atoi(argv[2]);
	}
	else {
		printf("Enter SMTP MTA and MDA ports!\n");
		exit(1);
	}

	serv_addr.sin_family = AF_INET;
	serv_addr.sin_addr.s_addr = INADDR_ANY;

	printf("SMPT Mailing System\n\n");

	char mail_id[100];
	printf("Enter your mail id: ");
	scanf("%s", mail_id);

	registerClient(strdup(mail_id), port2);

	int n;

	while(1) {
		printf("\n1. Send mail\n2. Receive mail\nEnter your command: ");
		scanf("%d", &n);

		if(n == 1) {
			serv_addr.sin_port = htons(port1);
			clientSend(serv_addr, mail_id);
		} else if(n == 2) {
			serv_addr.sin_port = htons(port2);
			clientRecv(serv_addr, mail_id);
		} else {
			printf("Invalid command!\n");
		}
	}
}
```

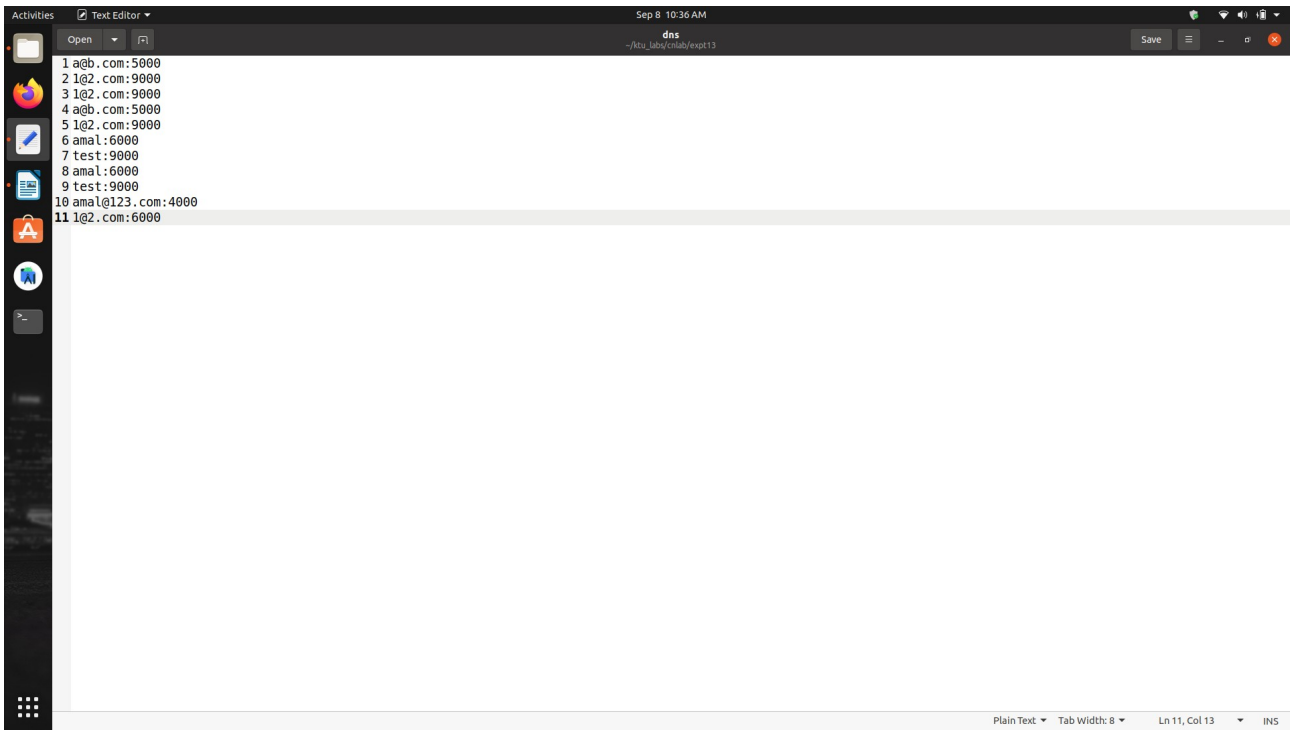
OUTPUT


Activities
Terminal
Sep 8 10:35 AM
amal@amal-TUF-Gaming-FX705DT-FX705DT: ~/ktu_labs/cnlab/expt13
amal@amal-TUF-Gaming-FX705DT-FX705DT:~/ktu_labs/cnlab/expt13$ ./mua 5000 4000
SMPT Mailing System
Enter your mail id: amal@123.com
Successfully registered mail id!
1. Send mail
2. Receive mail
Enter your command: 1
Enter "HELO" to initiate mail transfer: HELO
MAIL TO: 1@2.com
DATA:
Hey there!
Hope you are fine...
.
Mail successfully sent.
1. Send mail
2. Receive mail
Enter your command:


Activities
Terminal
Sep 8 10:35 AM
amal@amal-TUF-Gaming-FX705DT-FX705DT: ~/ktu_labs/cnlab/expt13
amal@amal-TUF-Gaming-FX705DT-FX705DT:~/ktu_labs/cnlab/expt13$ ./mua 7000 6000
SMPT Mailing System
Enter your mail id: 1@2.com
Successfully registered mail id!
1. Send mail
2. Receive mail
Enter your command: 2
Received mail!
MAIL FROM amal@123.com
RCP TO 1@2.com
DATA
Hey there!
Hope you are fine...
1. Send mail
2. Receive mail
Enter your command:


Activities
Terminal
Sep 8 10:35 AM
amal@amal-TUF-Gaming-FX705DT-FX705DT: ~/ktu_labs/cnlab/expt13
amal@amal-TUF-Gaming-FX705DT-FX705DT:~/ktu_labs/cnlab/expt13$ ./mta
SMTP MTA
Enter port: 5000
Server socket created.
Connected to client.
Sender: amal@123.com
Receiver: 1@2.com
Data:
Hey there!
Hope you are fine...
Forwarded to MDA 6000
Connection closed.


Activities
Terminal
Sep 8 10:35 AM
amal@amal-TUF-Gaming-FX705DT-FX705DT: ~/ktu_labs/cnlab/expt13
amal@amal-TUF-Gaming-FX705DT-FX705DT:~/ktu_labs/cnlab/expt13$ ./mda
SMTP MDA
Enter port: 6000
Server socket created.
Connected to client.
Mail received and queued!
Connected to client.
Queued mails sent to 1@2.com.


Activities
Text Editor
Sep 8 10:36 AM
Open
dns
~/ktu_labs/cnlab/expt13
Save
1 a@b.com:5000
2 1@2.com:9000
3 1@2.com:9000
4 a@b.com:5000
5 1@2.com:9000
6 amal:6000
7 test:9000
8 amal:6000
9 test:9000
10 amal@123.com:4000
11 1@2.com:6000
Plain Text
Tab Width: 8
Ln 11, Col 13
INS