**Code for server:**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <string.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <arpa/inet.h>

#include <netinet/in.h>

#define PORT 8080

#define MAXLINE 1024

// Function to determine the class of IPv4 address

char getClass(char \*ip)

{

char class;

int firstOctet = atoi(strtok(ip, "."));

if (firstOctet >= 1 && firstOctet <= 126)

{

class = 'A';

}

else if (firstOctet >= 128 && firstOctet <= 191)

{

class = 'B';

}

else if (firstOctet >= 192 && firstOctet <= 223)

{

class = 'C';

}

else if (firstOctet >= 224 && firstOctet <= 239)

{

class = 'D';

}

else

{

class = 'E';

}

return class;

}

// Driver code

int main()

{

int sockfd;

char buffer[MAXLINE];

struct sockaddr\_in servaddr, cliaddr;

int len, n;

// Creating socket file descriptor

if ((sockfd = socket(AF\_INET, SOCK\_DGRAM, 0)) < 0)

{

perror("socket creation failed");

exit(EXIT\_FAILURE);

}

memset(&servaddr, 0, sizeof(servaddr));

memset(&cliaddr, 0, sizeof(cliaddr));

// Filling server information

servaddr.sin\_family = AF\_INET; // IPv4

servaddr.sin\_addr.s\_addr = INADDR\_ANY;

servaddr.sin\_port = htons(PORT);

// Bind the socket with the server address

if (bind(sockfd, (const struct sockaddr \*)&servaddr, sizeof(servaddr)) < 0)

{

perror("bind failed");

exit(EXIT\_FAILURE);

}

len = sizeof(cliaddr); // len is value/result

n = recvfrom(sockfd, (char \*)buffer, MAXLINE, MSG\_WAITALL, (struct sockaddr \*)&cliaddr,&len);

buffer[n] = '\0';

// Determine class of IP address

char ip[MAXLINE];

strcpy(ip, buffer);

char ipClass = getClass(ip);

// Send class name back to client

sendto(sockfd, (const char \*)&ipClass, sizeof(ipClass), MSG\_CONFIRM, (const struct sockaddr\*)&cliaddr, len);

printf("Class of IP address %s is %c\n", ip, ipClass);

return 0;

}

**Code for client:**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <string.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <arpa/inet.h>

#include <netinet/in.h>

#define PORT 8080

#define MAXLINE 1024

// Driver code

int main()

{

int sockfd;

char buffer[MAXLINE];

char ip[MAXLINE];

struct sockaddr\_in servaddr;

// Creating socket file descriptor

if ((sockfd = socket(AF\_INET, SOCK\_DGRAM, 0)) < 0)

{

perror("socket creation failed");

exit(EXIT\_FAILURE);

}

memset(&servaddr, 0, sizeof(servaddr));

// Filling server information

servaddr.sin\_family = AF\_INET;

servaddr.sin\_port = htons(PORT);

printf("Enter an IPv4 address: ");

scanf("%s", ip);

// Send IPv4 address to server

sendto(sockfd, (const char \*)ip, strlen(ip), MSG\_CONFIRM, (const struct sockaddr \*)&servaddr,

sizeof(servaddr));

printf("IPv4 address sent.\n");

int n, len;

len = sizeof(servaddr);

n = recvfrom(sockfd, (char \*)buffer, MAXLINE, MSG\_WAITALL, (struct sockaddr\*)&servaddr, &len);

buffer[n] = '\0';

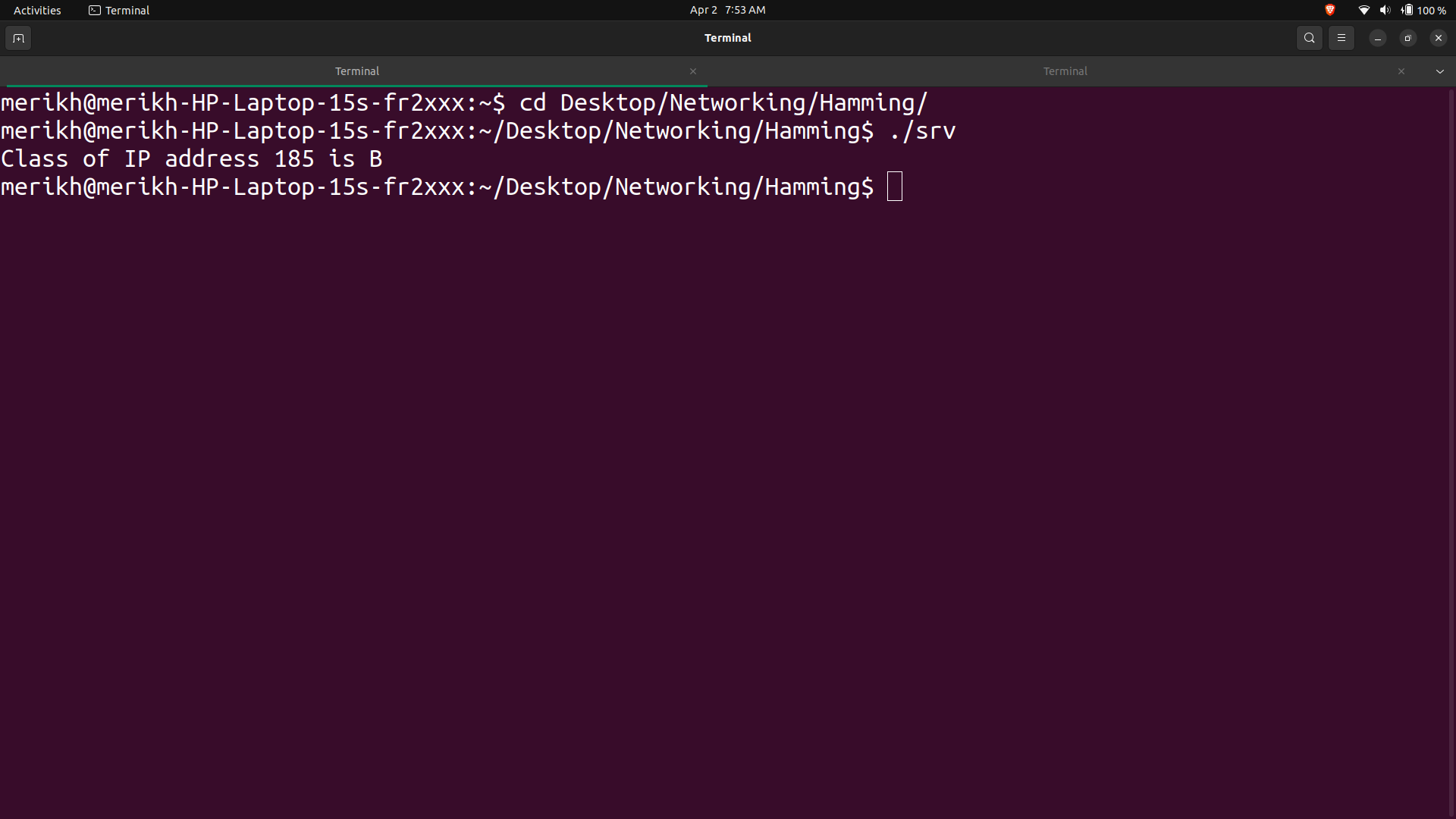
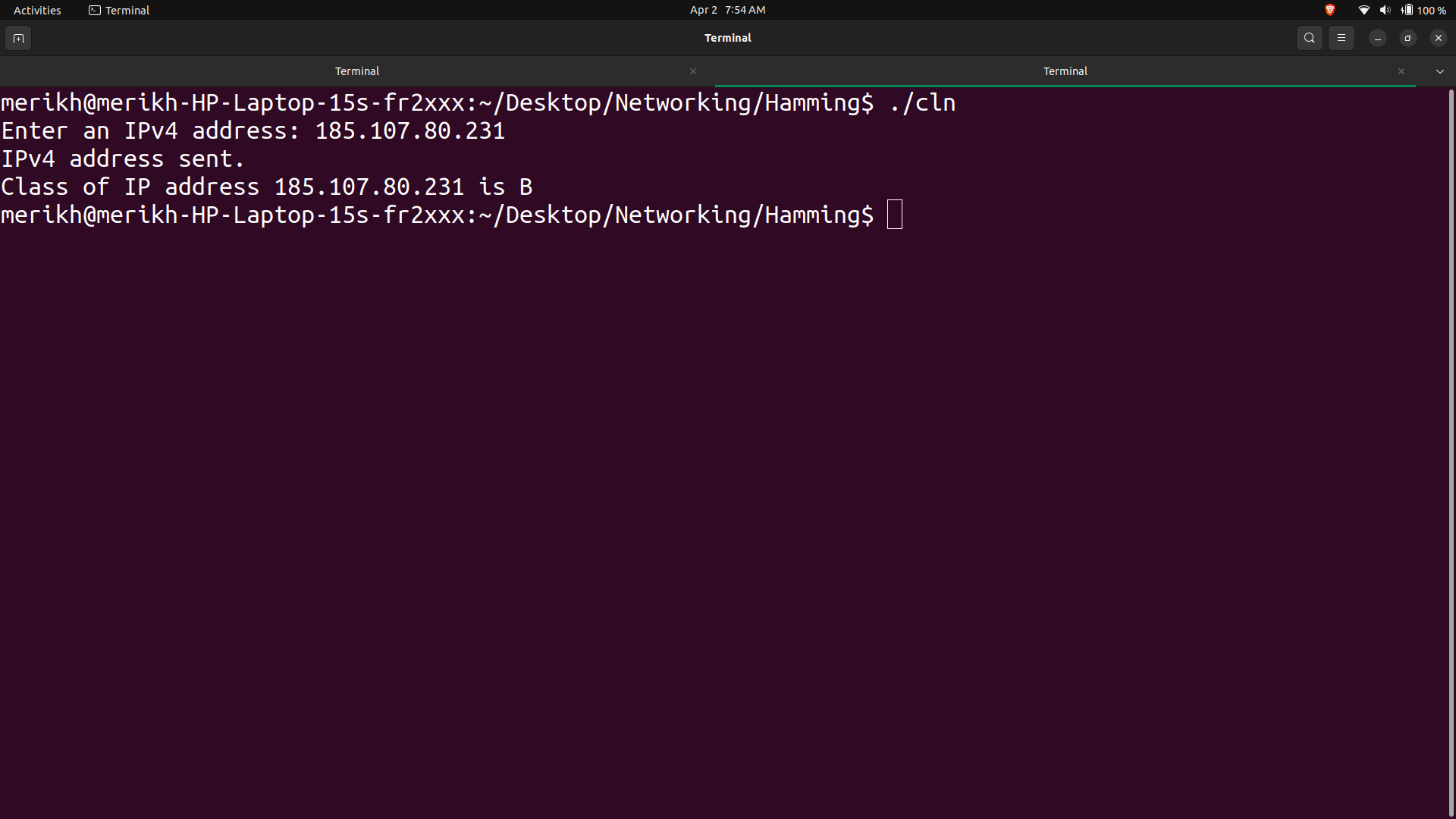
// Print class name received from server

printf("Class of IP address %s is %c\n", ip, buffer[0]);

close(sockfd);

return 0;

}

**Output:**