EXPERIMENT - 8

Name	Anumeya Sehgal
Registration Number	23BAI1203
Course Code	BCSE308P
Course Title	Computer Networks Lab
Date	10 October, 2024

AIM: To implement Classful and Classless IP Address classification in C

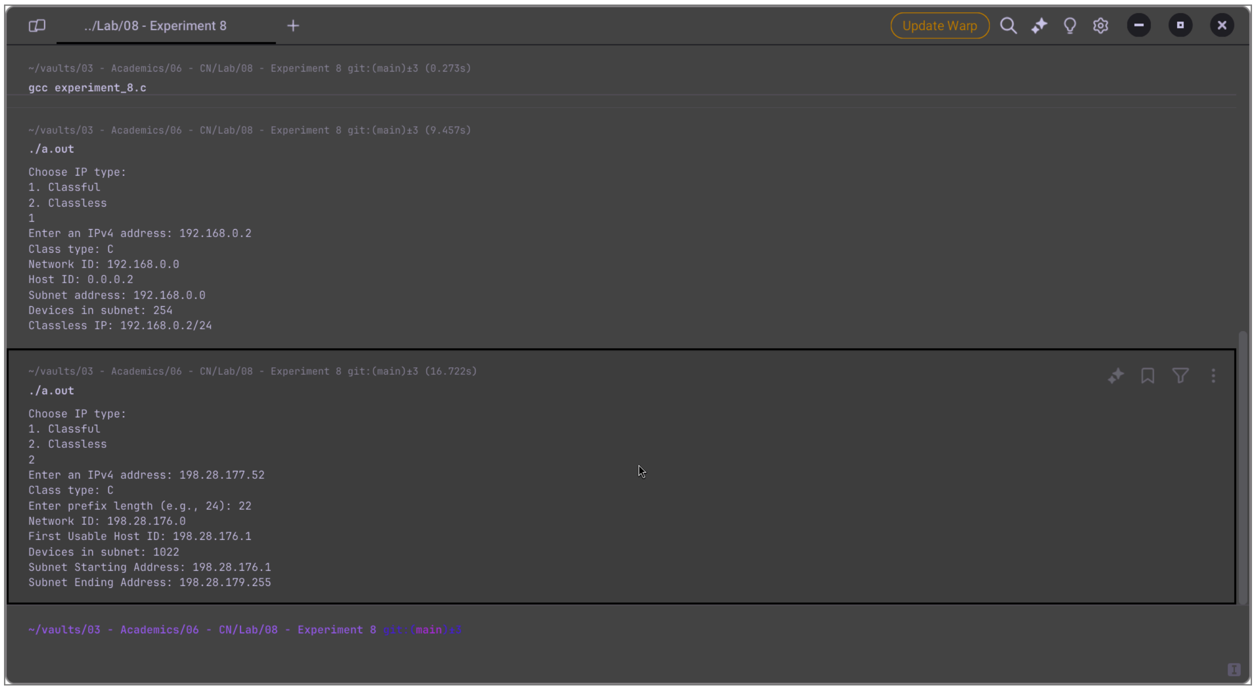
PROGRAM:
```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

void findClass(char *ip, char *classType) {
     int firstOctet = atoi(strtok(ip, "."));
     if (firstOctet ⩾ 1 && firstOctet ⩽ 126)
     strcpy(classType, "A");
     else if (firstOctet ⩾ 128 && firstOctet ⩽ 191)
     strcpy(classType, "B");
     else if (firstOctet ⩾ 192 && firstOctet ⩽ 223)
     strcpy(classType, "C");
     else if (firstOctet ⩾ 224 && firstOctet ⩽ 239)
     strcpy(classType, "D");
     else
     strcpy(classType, "E");
}

void findNetworkAndHostID(char *ip, char *classType, char
*networkID, char *hostID) {
     int octets[4];
     sscanf(ip, "%d.%d.%d.%d", &octets[0], &octets[1],
&octets[2], &octets[3]);

     if (strcmp(classType, "A") == 0) {
     sprintf(networkID, "%d.0.0.0", octets[0]);
     sprintf(hostID, "0.%d.%d.%d", octets[1], octets[2],
octets[3]);
     } else if (strcmp(classType, "B") == 0) {
     sprintf(networkID, "%d.%d.0.0", octets[0], octets[1]);
     sprintf(hostID, "0.0.%d.%d", octets[2], octets[3]);
     } else if (strcmp(classType, "C") == 0) {
     sprintf(networkID, "%d.%d.%d.0", octets[0], octets[1],
octets[2]);
     sprintf(hostID, "0.0.0.%d", octets[3]);
     } else {
     strcpy(networkID, "Not applicable");
     strcpy(hostID, "Not applicable");
     }
}

void findSubnetAddress(char *ip, char *classType, char
*subnetAddress) {
     int octets[4];
     sscanf(ip, "%d.%d.%d.%d", &octets[0], &octets[1],
&octets[2], &octets[3]);

     if (strcmp(classType, "A") == 0)
     sprintf(subnetAddress, "%d.0.0.0", octets[0]);
     else if (strcmp(classType, "B") == 0)
     sprintf(subnetAddress, "%d.%d.0.0", octets[0], octets[1]);
     else if (strcmp(classType, "C") == 0)
     sprintf(subnetAddress, "%d.%d.%d.0", octets[0], octets[1],
octets[2]);
     else
     strcpy(subnetAddress, "Not applicable");
```

```c
}

void findSubnetDevices(char *classType) {
     if (strcmp(classType, "A") == 0)
     printf("Devices in subnet: 16,777,214\n");
     else if (strcmp(classType, "B") == 0)
     printf("Devices in subnet: 65,534\n");
     else if (strcmp(classType, "C") == 0)
     printf("Devices in subnet: 254\n");
     else
     printf("Devices in subnet: Not applicable\n");
}

void findClasslessIP(char *ip, char *classType) {
     int prefixLength;
     if (strcmp(classType, "A") == 0)
     prefixLength = 8;  // Class A: /8
     else if (strcmp(classType, "B") == 0)
     prefixLength = 16; // Class B: /16
     else if (strcmp(classType, "C") == 0)
     prefixLength = 24; // Class C: /24
     else
     prefixLength = 0;  // Not applicable for Class D and E

     if (prefixLength > 0)
     printf("Classless IP: %s/%d\n", ip, prefixLength);
     else
     printf("Classless IP: Not applicable\n");
}

void calculateSubnetAddresses(char *ip, int prefixLength) {
     int octets[4];
     sscanf(ip, "%d.%d.%d.%d", &octets[0], &octets[1],
&octets[2], &octets[3]);
```

```
     int subnetMask = 0xFFFFFFFF << (32 - prefixLength);
     int networkAddress = (octets[0] << 24) | (octets[1] << 16)
| (octets[2] << 8) | octets[3];
     networkAddress &= subnetMask;

     int startAddress = networkAddress + 1; // First usable
address
     int endAddress = networkAddress | ~subnetMask; // Last
usable address

     printf("Subnet Starting Address: %d.%d.%d.%d\n",
          (startAddress >> 24) & 0xFF,
          (startAddress >> 16) & 0xFF,
          (startAddress >> 8) & 0xFF,
          startAddress & 0xFF);

     printf("Subnet Ending Address: %d.%d.%d.%d\n",
          (endAddress >> 24) & 0xFF,
          (endAddress >> 16) & 0xFF,
          (endAddress >> 8) & 0xFF,
          endAddress & 0xFF);
}

void calculateNetworkAndHostIDClassless(char *ip, int
prefixLength, char *networkID, char *hostID) {
     int octets[4];
     sscanf(ip, "%d.%d.%d.%d", &octets[0], &octets[1],
&octets[2], &octets[3]);

     // Calculate subnet mask
     int subnetMask = 0xFFFFFFFF << (32 - prefixLength);
     int networkAddress = (octets[0] << 24) | (octets[1] << 16)
| (octets[2] << 8) | octets[3];
     networkAddress &= subnetMask;
     // Set network ID
     sprintf(networkID, "%d.%d.%d.%d",
          (networkAddress >> 24) & 0xFF,
          (networkAddress >> 16) & 0xFF,
          (networkAddress >> 8) & 0xFF,
          networkAddress & 0xFF);

     // Calculate first usable host address
     int firstUsableHost = networkAddress + 1;
     sprintf(hostID, "%d.%d.%d.%d",
          (firstUsableHost >> 24) & 0xFF,
          (firstUsableHost >> 16) & 0xFF,
          (firstUsableHost >> 8) & 0xFF,
          firstUsableHost & 0xFF);
}

void findSubnetDevicesClassless(int prefixLength) {
     int numDevices = (1 << (32 - prefixLength)) - 2; // Total
usable addresses
     printf("Devices in subnet: %d\n", numDevices);
}

int main() {
     char ip[16], classType[2], networkID[16], hostID[16],
subnetAddress[16];
     int choice;

     printf("Choose IP type:\n1. Classful\n2. Classless\n");
     scanf("%d", &choice);

     printf("Enter an IPv4 address: ");
     scanf("%s", ip);

     char tempIp[16];
     strcpy(tempIp, ip);
```

```
    findClass(tempIp, classType);
    printf("Class type: %s\n", classType);

    if (choice == 1) {
    findNetworkAndHostID(ip, classType, networkID, hostID);
    printf("Network ID: %s\n", networkID);
    printf("Host ID: %s\n", hostID);

    findSubnetAddress(ip, classType, subnetAddress);
    printf("Subnet address: %s\n", subnetAddress);

    findSubnetDevices(classType);
    findClasslessIP(ip, classType);
    } else if (choice == 2) {
    int prefixLength;
    printf("Enter prefix length (e.g., 24): ");
    scanf("%d", &prefixLength);

    if (prefixLength < 1 || prefixLength > 30) {
         printf("Invalid prefix length. It should be between 1
and 30.\n");
         return 1;
    }

    calculateNetworkAndHostIDClassless(ip, prefixLength,
networkID, hostID);
    printf("Network ID: %s\n", networkID);
    printf("First Usable Host ID: %s\n", hostID);
    findSubnetDevicesClassless(prefixLength);
    calculateSubnetAddresses(ip, prefixLength);
    } else {
    printf("Invalid choice.\n");
    }
    return 0;
}
```

OUTPUT:



```
~/vaults/03 - Academics/06 - CN/Lab/08 - Experiment 8 git:(main)±3 (0.273s)
gcc experiment_8.c

~/vaults/03 - Academics/06 - CN/Lab/08 - Experiment 8 git:(main)±3 (9.457s)
./a.out
Choose IP type:
1. Classful
2. Classless
1
Enter an IPv4 address: 192.168.0.2
Class type: C
Network ID: 192.168.0.0
Host ID: 0.0.0.2
Subnet address: 192.168.0.0
Devices in subnet: 254
Classless IP: 192.168.0.2/24

~/vaults/03 - Academics/06 - CN/Lab/08 - Experiment 8 git:(main)±3 (16.722s)
./a.out
Choose IP type:
1. Classful
2. Classless
2
Enter an IPv4 address: 198.28.177.52
Class type: C
Enter prefix length (e.g., 24): 22
Network ID: 198.28.176.0
First Usable Host ID: 198.28.176.1
Devices in subnet: 1022
Subnet Starting Address: 198.28.176.1
Subnet Ending Address: 198.28.179.255

~/vaults/03 - Academics/06 - CN/Lab/08 - Experiment 8 git:(main)±3
```

```
./a.out
Choose IP type:
1. Classful
2. Classless
2
Enter an IPv4 address: 198.28.177.52
Class type: C
Enter prefix length (e.g., 24): 22
Network ID: 198.28.176.0
First Usable Host ID: 198.28.176.1
Devices in subnet: 1022
Subnet Starting Address: 198.28.176.1
Subnet Ending Address: 198.28.179.255
```

```
./a.out
Choose IP type:
1. Classful
2. Classless
1
Enter an IPv4 address: 192.168.0.2
Class type: C
Network ID: 192.168.0.0
Host ID: 0.0.0.2
Subnet address: 192.168.0.0
Devices in subnet: 254
Classless IP: 192.168.0.2/24
```