20BDS0146
VENNELA G

IP ADDRESSING

1. C program to check whether IP Address is Version 4 or Version 6

SOURCE CODE:

```
#include <stdio.h>
#include<string.h>
#include<stdbool.h>
#include<stdlib.h>
#include<ctype.h>
int validate_number(char *str) {
  while (*str)
  {
    if(!isdigit(*str)){ //if the character is not a number, return
      return false;
      return 0;
    }
    str++; //point to next character
  }
  return 1;
}

bool checkIPv4(char* s)
{
```

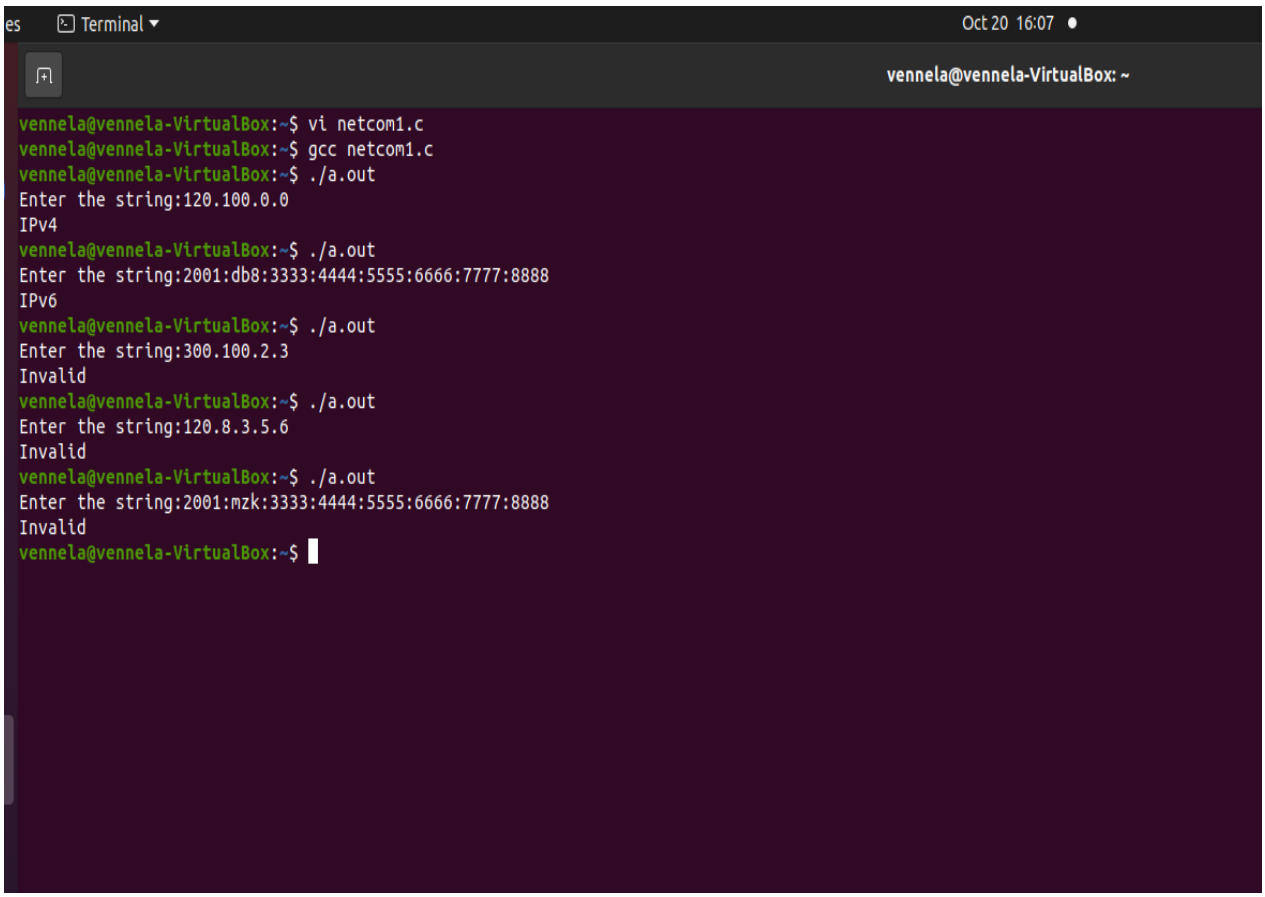
```
 //check whether the IP is valid or not
  int i, num, dots = 0;
  char *ptr;
  if (s == NULL)
    return 0;
    ptr = strtok(s, "."); //cut the string using dor delimiter
    if (ptr == NULL)
      return 0;
  while (ptr) {
    if (!validate_number(ptr)) //check whether the sub string is holding only
number or not
      return 0;
      num = atoi(ptr); //convert substring to number
      if (num >= 0 && num <= 255) {
        ptr = strtok(NULL, "."); //cut the next part of the string
        if (ptr != NULL)
          dots++; //increase the dot count
      } else
        return 0;
   }
   if (dots != 3) //if the number of dots are not 3, return false
     return 0;
    return 1;
}
// Function to check if the string
// represents a hexadecimal number
```

```
bool checkHex(char* s)
{

    int n = strlen(s);
   for (int i = 0; i < n; i++) {
        char ch = s[i];
        if ((ch < '0' || ch > '9' && ch!=':')
            && (ch < 'A' || ch > 'F')
            && (ch < 'a' || ch > 'f')) {
            return false;
  }

    }

    return true;}
bool checkIPv6(char* s)
{
    int cnt = 0;

    for (int i = 0; i < strlen(s)+1;i++) {
        if (s[i] == ':')
            cnt++;
    }

    // Not a valid IP Address
    if (cnt != 7)
```

```
            return false;



  checkHex(s);
}

// Function to check if the string
// S is IPv4 or IPv6 or Invalid
void checkIPAddress(char* s)
{
      // Check if the string is IPv4
      if (checkIPv4(s))
            printf("IPv4") ;

      // Check if the string is IPv6
      else if (checkIPv6(s))
          printf("IPv6");

      // Otherwise, print "Invalid"
      else
            printf("Invalid");
}

// Driver Code
int main()
{
```

```
    char s[100000];
    printf("Enter the string:");
    scanf("%[^\n]s",s);
    checkIPAddress(s);
    printf("\n");
    return 0;
}
```

OUTPUT:



2. C program to check class of IP Address version 4

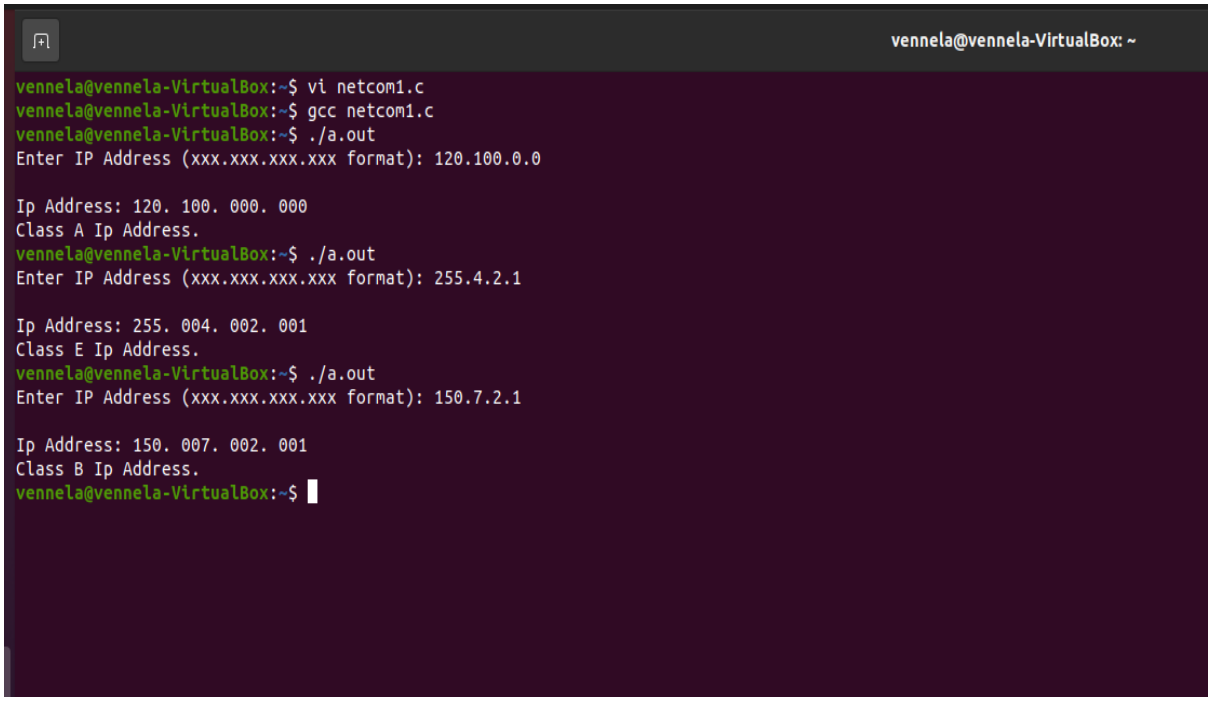
SOURCE CODE:

```
#include <stdio.h>
#include <string.h>
#include<stdlib.h>
void extractIpAddress(unsigned char *sourceString,short *ipAddress)
{
   unsigned short len=0;
   unsigned char  oct[4]={0},cnt=0,cnt1=0,i,buf[5];

   len=strlen(sourceString);
   for(i=0;i<len;i++)
   {
      if(sourceString[i]!='.'){
         buf[cnt++] =sourceString[i];
      }
      if(sourceString[i]=='.' || i==len-1){
         buf[cnt]='\0';
         cnt=0;
         oct[cnt1++]=atoi(buf);
      }
   }
   ipAddress[0]=oct[0];
   ipAddress[1]=oct[1];
```

```
    ipAddress[2]=oct[2];
    ipAddress[3]=oct[3];
}

int main()
{
    unsigned char ip[20]={0};
    short ipAddress[4];

    printf("Enter IP Address (xxx.xxx.xxx.xxx format): ");
    scanf("%s",ip);

    extractIpAddress(ip,&ipAddress[0]);

    printf("\nIp Address: %03d. %03d. %03d.
%03d\n",ipAddress[0],ipAddress[1],ipAddress[2],ipAddress[3]);

    if(ipAddress[0]>=0 && ipAddress[0]<=127)
        printf("Class A Ip Address.\n");
    if(ipAddress[0]>127 && ipAddress[0]<191)
        printf("Class B Ip Address.\n");
    if(ipAddress[0]>191 && ipAddress[0]<224)
        printf("Class C Ip Address.\n");
    if(ipAddress[0]>224 && ipAddress[0]<=239)
        printf("Class D Ip Address.\n");
    if(ipAddress[0]>239)
        printf("Class E Ip Address.\n");
```

```
    return 0;
}
```

OUTPUT:



3. C program to implement sub block allocation for requested group of customers

SOURCE CODE:

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<stdbool.h>
typedef struct
{
 int first,second,third,fourth;
 int mask;
}Address;
typedef struct{
 int no_cust;
 int add_per_cust;
 Address *start,*end;
}Group;
Address* readAndCreateAddress()
{
 Address *add=(Address*)malloc((sizeof(Address)));

scanf("%d%d%d%d%d",&(add->first),&(add->second),&(add->third),&(add-
>fourth),&(add->mask));
 return add;
}
```

```
void printAddress(Address *add)
{

printf("%d.%d.%d.%d/%d\n",add->first,add->second,add->third,add-
>fourth,add->mask);
}
Address * createAddress()
{
 Address *add=(Address*)malloc(sizeof(Address));
 return add;
}
void copyIP(Address* newIP,Address *IP)
{
 newIP->first=IP->first;
 newIP->second=IP->second;
 newIP->third=IP->third;
 newIP->fourth=IP->fourth;
 newIP->mask=IP->mask;
}
void incrementIP(Address *ip,int total)
{
 int counter=0;
 while(counter<total)
 { if(ip->fourth<255)
 {
 int x=255-ip->fourth;
 int increment=total-counter>x ? x: total-counter ;
```

```
counter+=increment;
ip->fourth+=increment;
}
else if(ip->third<255)
{
int x=255-ip->third;
int increment=1 ;
counter+=increment;
ip->third+=increment;
ip->fourth=0;
}
else if(ip->second<255)
{
int x=255-ip->second;
int increment=1 ;
counter+=increment;
ip->second+=increment;
ip->fourth=0;
ip->third=0;
}
else if(ip->first<255)
{
int x=255-ip->first;
int increment=1 ;
counter+=increment;
ip->first+=increment;
```

```
 ip->fourth=0;
 ip->third=0;
 ip->second=0;
 }
 }
 // printf("Total Address in this group: %d\n",counter);
 // printAddress(ip);
}
void distributeIP(Address *prev_hook,Address * START,Address *END,
unsigned long total, unsigned long* LIMIT)
{
 if(total>*LIMIT)
 {
 printf("OVERLIMIT!!!!!!\n");
 exit(0);
 }
 copyIP(START,prev_hook);
 copyIP(END,START);
 *LIMIT-=total;
 incrementIP(END,total-1);
 copyIP(prev_hook,END); incrementIP(prev_hook,1);
}
int main()
{
 printf("Welcome to ISP Addressing Solver 20BDS0146\n\n");
```

```
 printf("Do two favours :\n1.Please enter groups' details in descending order of
IP req. per customer !!!\n2.Enter only valid and possible
requirements!!!\n\n");


 printf("No need of any favours, this program is fully generalized for any valid
input!!!!!\n\n");


 Address *ISP;

 printf("Enter Network IP and Mask of ISP: ");

 ISP=readAndCreateAddress();

 unsigned long *LIMIT=(unsigned long*)malloc(sizeof(unsigned long));

 *LIMIT=(unsigned long)pow(2,32-ISP->mask);

 printf("The ISP has Network has [%lu] addresses starting with
address:",*LIMIT);

 printAddress(ISP);

 int n;

 printf("Enter the number of groups you want to create: ");

 scanf("%d",&n);

 Group **grps=(Group**)calloc(n,sizeof(Group*));

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

 {

 grps[i]=(Group*)malloc(sizeof(Group));

 printf("Enter the number of customers and addresses_addresses per customer
for Group[%d] : ",(i+1));

 scanf("%d%d",&grps[i]->no_cust,&grps[i]->add_per_cust);

 }

 Address *prev_add=createAddress();

 copyIP(prev_add,ISP);
```

```
for(int i=0;i<n;i++)
{
int x=0;
while((int)pow(2,x)<grps[i]->add_per_cust) x++;
grps[i]->start=createAddress();
grps[i]->end=createAddress();
grps[i]->start->mask=32-x;
grps[i]->end->mask=32-x;
prev_add->mask=32-x;
distributeIP(prev_add,grps[i]->start,grps[i]->end,(unsigned
long )grps[i]->no_cust*grps[i]->add_per_cust,LIMIT);
}
for(int i=0;i<n;i++) {
printf("-----Details of Group[%d]-----\n",(i+1));
printf("Starting IP Address: ");
printAddress(grps[i]->start);
printf("Last IP Address: ");
printAddress(grps[i]->end);
printf("Total IP addresses: %lu\n",((unsigned
long)grps[i]->no_cust*(unsigned long)grps[i]->add_per_cust));
printf("-----------------------------------\n\n");
}
int cus;
Address *temp=prev_add;
for(int i=0;i<n;i++)
{
```

```
 printf("Enter customer no of Group[%d]: ",(i+1));
 scanf("%d",&cus);
 copyIP(temp,grps[i]->start);
 incrementIP(temp,(unsigned long)(cus-1)*grps[i]->add_per_cust);
 printf("-------------\nStarting IP of Customer [%d]: ",cus);
 printAddress(temp);
 incrementIP(temp,(unsigned long)(grps[i]->add_per_cust-1));
 printf("Ending IP of Customer [%d]: ",cus);
 printAddress(temp);
 printf("-----------------------------\n\n");
 }
 return 0;
}
```

OUTPUT:

```
vennela@vennela-VirtualBox: ~
vennela@vennela-VirtualBox:~$ gcc netcom1.c -lm
vennela@vennela-VirtualBox:~$ ./a.out
Welcome to ISP Addressing Solver 20BDS0146

Do two favours :
1.Please enter groups' details in descending order of IP req. per customer !!!
2.Enter only valid and possible requirements!!!

No need of any favours, this program is fully generalized for any valid input!!!!!

Enter Network IP and Mask of ISP: 120 100 0 0 16
The ISP has Network has [65536] addresses starting with address:120.100.0.0/16
Enter the number of groups you want to create: 3
Enter the number of customers and addresses_addresses per customer for Group[1] : 64 256
Enter the number of customers and addresses_addresses per customer for Group[2] : 128 128
Enter the number of customers and addresses_addresses per customer for Group[3] : 128 64
-----Details of Group[1]-----
Starting IP Address: 120.100.0.0/24
Last IP Address: 120.100.63.255/24
Total IP addresses: 16384
-----------------------------------

-----Details of Group[2]-----
Starting IP Address: 120.100.64.0/25
Last IP Address: 120.100.127.255/25
Total IP addresses: 16384
-----------------------------------

-----Details of Group[3]-----
Starting IP Address: 120.100.128.0/26
Last IP Address: 120.100.159.255/26
Total IP addresses: 8192
-----------------------------------

Enter customer no of Group[1]: 64
-------------
Starting IP of Customer [64]: 120.100.63.0/24
Ending IP of Customer [64]: 120.100.63.255/24
----------------------------

Enter customer no of Group[2]: 128
-------------
Starting IP of Customer [128]: 120.100.127.128/25
Ending IP of Customer [128]: 120.100.127.255/25
----------------------------

Enter customer no of Group[3]: 128
-------------
```

```
vennela@vennela-VirtualBox: ~
1.Please enter groups' details in descending order of IP req. per customer !!!
2.Enter only valid and possible requirements!!!

No need of any favours, this program is fully generalized for any valid input!!!!!

Enter Network IP and Mask of ISP: 120 100 0 0 16
The ISP has Network has [65536] addresses starting with address:120.100.0.0/16
Enter the number of groups you want to create: 3
Enter the number of customers and addresses_addresses per customer for Group[1] : 64 256
Enter the number of customers and addresses_addresses per customer for Group[2] : 128 128
Enter the number of customers and addresses_addresses per customer for Group[3] : 128 64
-----Details of Group[1]-----
Starting IP Address: 120.100.0.0/24
Last IP Address: 120.100.63.255/24
Total IP addresses: 16384
-----------------------------------

-----Details of Group[2]-----
Starting IP Address: 120.100.64.0/25
Last IP Address: 120.100.127.255/25
Total IP addresses: 16384
-----------------------------------

-----Details of Group[3]-----
Starting IP Address: 120.100.128.0/26
Last IP Address: 120.100.159.255/26
Total IP addresses: 8192
-----------------------------------

Enter customer no of Group[1]: 64
-------------
Starting IP of Customer [64]: 120.100.63.0/24
Ending IP of Customer [64]: 120.100.63.255/24
----------------------------

Enter customer no of Group[2]: 128
-------------
Starting IP of Customer [128]: 120.100.127.128/25
Ending IP of Customer [128]: 120.100.127.255/25
----------------------------

Enter customer no of Group[3]: 128
-------------
Starting IP of Customer [128]: 120.100.159.192/26
Ending IP of Customer [128]: 120.100.159.255/26
----------------------------

vennela@vennela-VirtualBox:~$
```