Aim: Write a C program to implement classful addressing

Program:

```
#include<stdio.h>
#include<string.h>
char findClass(char str[])
{
	char arr[4];
	int i = 0;
	while (str[i] != '.')
	{
		arr[i] = str[i];
		i++;
	}
	i--;
	int ip = 0, j = 1;
	while (i >= 0)
	{
		ip = ip + (str[i] - '0') * j;
		j = j * 10;
		i--;
	}
	if (ip >=1 && ip <= 126)
		return 'A';
	else if (ip >= 128 && ip <= 191)
		return 'B';
	else if (ip >= 192 && ip <= 223)
		return 'C';
	else if (ip >= 224 && ip <= 239)
		return 'D';
	else
		return 'E';
}
void separate(char str[], char ipClass)
{
	char network[12], host[12];
	for (int k = 0; k < 12; k++){
		network[k] = '\0';
		host[k] = '\0';
	}
	if (ipClass == 'A'){
		int i = 0, j = 0;
		while (str[j] != '.')
			network[i++] = str[j++];
		i = 0;
		j++;
		while (str[j] != '\0')
			host[i++] = str[j++];
		printf("Network ID is %s\n", network);
		printf("Host ID is %s\n", host);
	}
```

```
	else if (ipClass == 'B')
	{
		int i = 0, j = 0, dotCount = 0;
		while (dotCount < 2)
		{
			network[i++] = str[j++];
			if (str[j] == '.')
				dotCount++;
		}
		i = 0;
		j++;
		while (str[j] != '\0')
			host[i++] = str[j++];
		printf("Network ID is %s\n", network);
		printf("Host ID is %s\n", host);
	}
	else if (ipClass == 'C')
	{
		int i = 0, j = 0, dotCount = 0;
		while (dotCount < 3)
		{
			network[i++] = str[j++];
			if (str[j] == '.')
				dotCount++;
		}
		i = 0;
		j++;
		while (str[j] != '\0')
			host[i++] = str[j++];
		printf("Network ID is %s\n", network);
		printf("Host ID is %s\n", host);
	}
	else
		printf("In this Class, IP address is not divided into Network and Host ID\n");
}
int main()
{
	char str[] = "192.226.12.11";
	char ipClass = findClass(str);
	printf("Given IP address belongs to Class %c\n",ipClass);
	separate(str, ipClass);
	return 0;
}
```

Output:

```
Given IP address belongs to Class C
Network ID is 192.226.12
Host ID is 11
```

Aim: Write a C program to implement classless addressing

Program:

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

char *ip_classless(char *ip_address) {
	int i, j, k;
	int octets[4];
	char *mask = (char *) malloc(sizeof(char) * 16);
	int bits = 0;
	for (i = 0, j = 0, k = 0; i < strlen(ip_address); i++) {
		if (ip_address[i] == '.') {
			octets[j++] = atoi(ip_address + k);
			k = i + 1;
		}
	}
	octets[j] = atoi(ip_address + k);
	for (i = 0; i < 4; i++) {
		int octet = octets[i];
		for (j = 7; j >= 0; j--) {
			if (octet >= (1 << j)) {
				bits++;
				octet -= (1 << j);
			} else if (bits % 8 != 0) {
				break;
			}
		}
	}
	sprintf(mask, "%s/%d", ip_address, bits);
	return mask;
}

int main() {
	char ip_address[16]="192.168.0.0";
	printf("Classless address: %s\n", ip_classless(ip_address));
	return 0;
}
```

Output:

```
Classless address: 192.168.0.0/3
```

Aim: Write a C program to implement Link State Routing

Program:

```
#include <stdio.h>
int main() {
  int count, src_router, i, j, k, w, v, min;
  int cost_matrix[100][100], dist[100], last[100];
  int flag[100];
  printf("\n Enter the no of routers");
  scanf("%d", &count);
  printf("\n Enter the cost matrix values:");
  for (i = 0; i < count; i++) {
    for (j = 0; j < count; j++) {
      scanf("%d", &cost_matrix[i][j]);
      if (cost_matrix[i][j] < 0) cost_matrix[i][j] = 1000;
    }
  }
  printf("\n Enter the source router:");
  scanf("%d", &src_router);
  for (v = 0; v < count; v++)
  {
    flag[v] = 0;
    last[v] = src_router;
    dist[v] = cost_matrix[src_router][v];
  }
  flag[src_router] = 1;
  for (i = 0; i < count; i++)
  {
    min = 1000;
    for (w = 0; w < count; w++)
    {
      if (!flag[w])
      {
        if (dist[w] < min)
        {
          v = w;
          min = dist[w];
        }
      }
    }
    flag[v] = 1;
    for (w = 0; w < count; w++)
    {
      if (!flag[w])
        if (min + cost_matrix[v][w] < dist[w])
        {
          dist[w] = min + cost_matrix[v][w];
          last[w] = v;
        }
    }
  }
```

```
  for (i = 0; i < count; i++)
  {
    printf("\n%d==>%d:Path taken:%d", src_router, i, i);
    w = i;
    while (w != src_router)
    {
      printf("<--%d", last[w]);
      w = last[w];
    }
    printf("\n Shortest path cost:%d", dist[i]);
  }
}
```

Output:

```
Enter the no of routers4

 Enter the cost matrix values:0 5 10 0
5 0 3 11
10 3 0 2
0 11 2 0

 Enter the source router:1

1==>0: Path taken:0<--1
 Shortest path cost:5
1==>1: Path taken:1
 Shortest path cost:0
1==>2: Path taken:2<--1
 Shortest path cost:3
1==>3: Path taken:3<--2<--1
 Shortest path cost:5
```

Aim: Implement data encryption and data decryption.

Program:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void main() {
   int i, ch, lp;
   char cipher[50], plain[50],key[50];
   while (1) {
      printf("\n-----MENU ----\n");
      printf("\n1:Data Encryption\t\n\n2:Data Decryption\t\n\n3:Exit");
      printf("\n\nEnter your choice:");
      scanf("%d", &ch);
```

```
    switch (ch) {
      case 1:
        printf("\nData Encryption");
        printf("\nEnter the plain text:");
        fflush(stdin);
        scanf("%s", &plain);
        printf("\nEnter the encryption key:");
        scanf("%s", &key);
        lp = strlen(key);
        for (i = 0; plain[i] != '\0'; i++)
          cipher[i] = plain[i] ^ lp;
        cipher[i] = '\0';
        printf("\nThe encrypted text is:");
        puts(cipher);break;
      case 2:
        printf("\nData decryption");
        for (i = 0; cipher[i] != '\0'; i++) plain[i] = cipher[i] ^ lp;
        printf("\nDecrypted text is:");
        puts(plain);
        break;
      case 3:
        exit(0);
    }}}
```

Output:

```
-----MENU ----

1:Data Encryption

2:Data Decryption

3:Exit

Enter your choice:1

Data Encryption
Enter the plain text:Mgit

Enter the encryption key:213

The encrypted text is:Ndjw
```

Aim: write a program to divide a given network into n-sub networks.

Program:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void subnetting(char *ip, int n) {
   // Convert IP address to binary
   unsigned int ipAddress;
   scanf(ip, "%u", &ipAddress);
   unsigned int subnetMask = (1U << 32) - 1;
   subnetMask <<= (32 - n);
   unsigned int hostsPerSubnet = (1U << (32 - n)) - 2;
   unsigned int subnetBase = ipAddress & subnetMask;
   printf("Subnet Mask: %u\n", subnetMask);
   printf("Number of Hosts per Subnet: %u\n", hostsPerSubnet);
   for (int i = 0; i < n; i++) {
      unsigned int subnetStart = subnetBase + i * (1U << (32 - n));
      unsigned int subnetEnd = subnetStart + (1U << (32 - n)) - 1;
      printf("\nSubnet %d:\n", i + 1);
      printf("Subnet Address Range: %u - %u\n", subnetStart, subnetEnd);
   }
}
int main() {
   char ipAddress[16]; //IPv4 address
   int numSubnets;
   printf("Enter IP address: ");
   scanf("%15s", ipAddress);
   printf("Enter the number of subnets: ");
   scanf("%d", &numSubnets);
   if (numSubnets <= 0 || numSubnets > 32) {
      printf("Invalid number of subnets. Please enter a value between 1 and 32.\n");
      return 1;
   }
   subnetting(ipAddress, numSubnets);
   return 0;
}
```

Output:

```
Enter IP address: 128.208.0.0
Enter the number of subnets: 2
Subnet Mask: 3221225472
Number of Hosts per Subnet: 1073741822

Subnet 1:
Subnet Address Range: 0 - 1073741823

Subnet 2:
Subnet Address Range: 1073741824 - 2147483647
```