CN Programs from 1 – 9

1. Implement the data link layer framing methods such as character, character-stuffing and bit stuffing.

1a. Character stuffing

Program

```
// Implement the data link layer framing methods such as, character-stuffing and bit
// 1a. Character Stuffing
#include<stdio.h>
void
main()
{
char c[50],d[50],t[50];
int i,m,j;
printf("Enter the number of characters : ");
scanf("%d",&m);
getchar();
printf("\nEnter the characters : ");
for(i=0;i<m;i++)
{
scanf(" %c",&c[i]);
}
printf("\nOriginal data : ");
for(i=0;i<m+1;i++)
printf("%c",c[i]);

d[0]='d';
d[1]='l';
d[2]='e';
d[3]='s';
d[4]='t';
```

```
d[5]='x';
for(i=0,j=6;i<m+1;i++,j++)
{
if((c[i]=='d'&&c[i+1]=='l'&&c[i+2]=='e'))
{
d[j]='d';
j++;
d[j]='l';
j++;
d[j]='e';
j++;
m=m+3;
}
d[j]=c[i];
}
m=m+6;
m++;
d[m]='d';
m++;
d[m]='l';
m++;
d[m]='e';
m++;
d[m]='e';
m++;
d[m]='t';
m++;
d[m]='x';
m++;
printf("\n\nTransmitted data : ");
for(i=0;i<m;i++)
```

```
{
printf("%c",d[i]);
}
for(i=6,j=0;i<m-6;i++,j++)
{
if(d[i]=='d'&&d[i+1]=='l'&&d[i+2]=='e'&&d[i+3]=='d'&&d[i+4]=='l'&&d[i+5]=='e')
i=i+3;
t[j]=d[i];
}
printf("\n\nReceived data : ");
for(i=0;i<j;i++){
printf("%c",t[i]);
}  }
```

Output:

```
Enter the number of characters : 12
Enter the characters : MrsdleAlekya
Original data : MrsdleAlekya
Transmitted data : dlestxMrsdledleAlekya dleetx
Received data : MrsdleAlekya
========================
```

1b. Bit stuffing

```
//1b. bit stuffing
#include<stdio.h>
void main()
{
        int a[15];
        int i, j, k, n, c=0, pos=0;
        printf("Enter the number of bits : ");
        scanf("%d",&n);
```

```
	printf("Enter the bits in 0s and 1s : ");
	for(i=0;i<n;i++)
		scanf("%d",&a[i]);
	for(i=0;i<n;i++)
	{
		if(a[i]==1)
		{
			c++;
			if(c==5)
			{
				pos=i+1;
				c=0;
				for(j=n-1;j>=pos;j--)
				{
					k=j+1;
					a[k]=a[j];
				}
				a[pos]=0;
				n=n+1;
			}
		}
		else
		c=0;
	}
	printf("Data after stuffing : ");
	printf("01111110 ");

	for(i=0;i<n;i++)
		printf("%d",a[i]);
	printf(" 01111110");
}
```

Output:

Enter the number of bits : 10

Enter the bits in 0s and 1s : 1 0 0 1 1 1 1 1 0 1

Data after stuffing : 01111110 10011111001 01111110

==========

2. Write a program to compute CRC code for the polynomials CRC-12, CRC-16 and CRC CCIP

Program

```
#include<stdio.h>
#include<string.h>
#define n strlen(g)

char t[28],cs[28],g[28];
int a,e,c,b;

void xor()
{
for(c=1;c<n;c++)
cs[c]=((cs[c]==g[c])?'0':'1');
}

void crc()
{
for(e=0;e<n;e++)
cs[e]=t[e];
do
{ if(cs[0]=='1')
xor();
cs[c]=cs[c+1];
```

```
cs[c]=t[e++];
}while(e<=a+n-1);
}

int main()
{
int flag=0;
do{
printf("\n1.crc12\n2.crc16\ncrc ccit\n4.exit\n\nEnter your option.");
scanf("%d",&b);
switch(b)
{ case 1:strcpy(g,"1100000001111");
break;
case 2:strcpy(g,"11000000000000101");
break;
case 3:strcpy(g,"10001000000100001");
break;
case 4:return 0;
}
printf("\n enter data:");
scanf("%s",t);
printf("\n----------------------\n");
printf("\n generating polynomial:%s",g);
a=strlen(t);
for(e=a;e<a+n-1;e++)
t[e]='0';
printf("\n-------------------------\n");
printf("modified data is:%s",t);
printf("\n----------------------\n");
crc();
printf("checksum is:%s",cs);
```

```
for(e=a;e<a+n-1;e++)
t[e]=cs[e-a];
printf("\n-----------------------\n");
printf("\n final codeword is : %s",t);
printf("\n-----------------------\n");
printf("\ntest error detection 0(yes) 1(no)?:");
scanf("%d",&e);
if(e==0)
{
do{
printf("\n\tenter the position where error is to be inserted:");
scanf("%d",&e);
}
while(e==0||e>a+n-1);
t[e-1]=(t[e-1]=='0')?'1':'0';
printf("\n-----------------------\n");
printf("\n\tErroneous data:%s\n",t);
}
crc();
for(e=0;(e<n-1)&&(cs[e]!='1');e++);
if(e<n-1)
printf("Error detected\n\n");
else
printf("\n No error detected \n\n");
printf("\n-----------------------");
}while(flag!=1);
}
```

Output

1.crc12

2.crc16

crc ccit

4.exit

Enter your option.1

enter data:10101101

---

generating polynomial:1100000001111

---

modified data is:10101101000000000000

---

checksum is:1010110100000

---

final codeword is : 10101101101011010000

---

test error detection 0(yes) 1(no)?:0

enter the position where error is to be inserted:3

Erroneous data:10001101101011010000

Error detected

1.crc12

2.crc16

crc ccit

4.exit

Enter your option.4

==========

3. Develop a simple data link layer that performs the flow control using the sliding window protocol, and loss recovery using the Go-Back-N mechanism.

Program

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

```
#include <stdlib.h>
#include <time.h>

void transmission(int *i, int *N, int *frames, int *total) {
    int k;
    while (*i <= *frames) {
        int z = 0;
        for (k = *i; k < *i + *N && k <= *frames; k++) {
            printf("Sending Frame %d...\n", k);
            (*total)++;
        }
        for (k = *i; k < *i + *N && k <= *frames; k++) {
            int f = rand() % 2;
            if (!f) {
                printf("Acknowledgment for Frame %d...\n", k);
                z++;
            } else {
                printf("Timeout!! Frame Number: %d Not Received\n", k);
                printf("Retransmitting Window...\n");
                break;
            }
        }
        printf("\n");
        *i = *i + z;
    }
}

int main() {
    int frames, N, total = 0;
    srand(time(NULL));
    printf("Enter the Total number of frames: ");
```

```
    scanf("%d", &frames);
    printf("Enter the Window Size: ");
    scanf("%d", &N);
    int i = 1;
    transmission(&i, &N, &frames, &total);
    printf("Total number of frames which were sent and resent are: %d\n", total);
    return 0;
}
```

Output

```
Enter the Total number of frames:
8
Enter the Window Size: 5
Sending Frame 1...
Sending Frame 2...
Sending Frame 3...
Sending Frame 4...
Sending Frame 5...
Acknowledgment for Frame 1...
Acknowledgment for Frame 2...
Acknowledgment for Frame 3...
Timeout!! Frame Number: 4 Not Received
Retransmitting Window...

Sending Frame 4...
Sending Frame 5...
Sending Frame 6...
Sending Frame 7...
Sending Frame 8...
Acknowledgment for Frame 4...
Acknowledgment for Frame 5...
```

Acknowledgment for Frame 6...

Acknowledgment for Frame 7...

Acknowledgment for Frame 8...

Total number of frames which were sent and resent are: 10

==========

4. Implement Dijsktra's algorithm to compute the shortest path through a network

Program

```
#include <stdio.h>
#define INFINITY 9999
#define max 10

void dijkstra(int G[max][max], int n, int startnode)
{
   int cost[max][max], distance[max], pred[max];
   int visited[max], count, mindistance, nextnode, i, j;

   for (i = 0; i < n; i++)
      for (j = 0; j < n; j++)
         if (G[i][j] == 0)
            cost[i][j] = INFINITY;
         else
            cost[i][j] = G[i][j];

   for (i = 0; i < n; i++) {
      distance[i] = cost[startnode][i];
      pred[i] = startnode;
      visited[i] = 0;
   }
```

```
distance[startnode] = 0;
visited[startnode] = 1;
count = 0;

while (count < n) {
    mindistance = INFINITY;
    nextnode = -1; // Initialize to invalid value

    for (i = 0; i < n; i++) {
        if (distance[i] < mindistance && !visited[i]) {
            mindistance = distance[i];
            nextnode = i;
        }
    }

    if (nextnode == -1)
        break; // No more reachable nodes

    visited[nextnode] = 1;

    for (i = 0; i < n; i++) {
        if (!visited[i] && mindistance + cost[nextnode][i] < distance[i]) {
            distance[i] = mindistance + cost[nextnode][i];
            pred[i] = nextnode;
        }
    }

    count++;
}
```

```
    for (i = 0; i < n; i++)
        if (i != startnode) {
            printf("\nDistance of node %d = %d", i + 1, distance[i]);
            printf("\nPath = %d", i + 1);
            j = i;
            do {
                j = pred[j];
                printf(" <- %d", j + 1);
            } while (j != startnode);
        }
}

int main()
{
    int ord, i, j, u, G[max][max];
    printf("Enter the required order of the matrix:");
    scanf("%d", &ord);
    printf("Enter the elements of the matrix:\n");
    for (i = 0; i < ord; i++)
        for (j = 0; j < ord; j++)
            scanf("%d", &G[i][j]);
    printf("Enter where you want to start (1-%d):", ord);
    scanf("%d", &u);
    dijkstra(G, ord, u - 1);
    return 0;
}
```

Output

```
Enter the required order of the matrix:4
Enter the elements of the matrix:
0       2       999     4
```

```
2       0       5       9999
9999    5       0       6
4       9999    6       0
Enter where you want to start (1-4):1

Distance of node 2 = 2
Path = 2 <- 1
Distance of node 3 = 7
Path = 3 <- 2 <- 1
Distance of node 4 = 4
Path = 4 <- 1
```

==========

5. Take an example subnet of hosts and obtain a broadcast tree for the subnet.

Program

```
#include <stdio.h>

#define N 5 // Number of hosts (nodes) in the subnet

void broadcastTree(int adjacencyMatrix[N][N], int root) {
    int queue[N];
    int front = -1, rear = -1,i;
    int visited[N] = {0};

    queue[++rear] = root;
    visited[root] = 1;

    printf("Broadcast tree for the subnet:\n");
    printf("Root node: %d\n", root);
```

```
    while (front < rear) {
        int currentNode = queue[++front];
        printf("Node %d -> ", currentNode);

        for (i = 0; i < N; i++) {
            if (adjacencyMatrix[currentNode][i] && !visited[i]) {
                queue[++rear] = i;
                visited[i] = 1;
            }
        }
    }
    printf("END\n");
}

int main() {
    int adjacencyMatrix[N][N] = {
        {0, 1, 1, 0, 0},
        {1, 0, 1, 1, 0},
        {1, 1, 0, 1, 1},
        {0, 1, 1, 0, 1},
        {0, 0, 1, 1, 0}
    };

    int root = 0; // The root node for the broadcast tree

    broadcastTree(adjacencyMatrix, root);

    return 0;
}
```

Output:

Broadcast tree for the subnet:

Root node: 0

Node 0 -> Node 1 -> Node 2 -> Node 3 -> Node 4 -> END

==========

6. Implement distance vector routing algorithm for obtaining routing tables at each node.

Program

```
#include <stdio.h>
struct node {
   int dist[20];
   int from[20];
} route[10];

int main() {
   int dm[20][20], no;
   int i, j, k; // Move the variable declarations outside the for loop.

   printf("Enter no of nodes: ");
   scanf("%d", &no);
   printf("Enter the distance matrix:\n");
   for (i = 0; i < no; i++) {
      for (j = 0; j < no; j++) {
         scanf("%d", &dm[i][j]);
         /* Set distance from i to i as 0 */
         dm[i][i] = 0;
         route[i].dist[j] = dm[i][j];
         route[i].from[j] = j;
      }
   }
```

```
    int flag;
    do {
        flag = 0;
        for (i = 0; i < no; i++) {
            for (j = 0; j < no; j++) {
                for (k = 0; k < no; k++) {
                    if ((route[i].dist[j]) > (route[i].dist[k] + route[k].dist[j])) {
                        route[i].dist[j] = route[i].dist[k] + route[k].dist[j];
                        route[i].from[j] = k;
                        flag = 1;
                    }
                }
            }
        }
    } while (flag);

    for (i = 0; i < no; i++) {
        printf("\nRouter info for router: %d\n", i + 1);
        printf("Dest\tNext Hop\tDist\n");
        for (j = 0; j < no; j++)
            printf("%d\t%d\t\t%d\n", j + 1, route[i].from[j] + 1, route[i].dist[j]);
    }
    return 0;
}
```

Output:

Enter no of nodes: 3

Enter the distance matrix:

0 1 1

1 0 1

1 1 0

Router info for router: 1

Dest	Next Hop	Dist
1	1	0
2	2	1
3	3	1

Router info for router: 2

Dest	Next Hop	Dist
1	1	1
2	2	0
3	3	1

Router info for router: 3

Dest	Next Hop	Dist
1	1	1
2	2	1
3	3	0

==========

7. Implement data encryption and data decryption.

Program

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

```
long int p, q, n, t, flag, e[100], d[100], temp[100], j, m[100], en[100], i;
char msg[100];
int prime(long int);
void ce();
long int cd(long int);
void encrypt();
void decrypt();
void main()
{
   printf("\nENTER FIRST PRIME NUMBER\n");
   scanf("%d", &p);
   flag = prime(p);
   if (flag == 0)
   {
      printf("\nWRONG INPUT\n");
      getch();
      exit(1);
   }
   printf("\nENTER ANOTHER PRIME NUMBER\n");
   scanf("%d", &q);
   flag = prime(q);
   if (flag == 0 || p == q)
   {
      printf("\nWRONG INPUT\n");
      getch();
      exit(1);
   }
   printf("\nENTER MESSAGE\n");
   fflush(stdin);
   scanf("%s", msg);
```

```
    for (i = 0; msg[i] != NULL; i++)
        m[i] = msg[i];
    n = p * q;
    t = (p - 1) * (q - 1);
    ce();
    printf("\nPOSSIBLE VALUES OF e AND d ARE\n");
    for (i = 0; i < j - 1; i++)
        printf("\n%ld\t%ld", e[i], d[i]);
    encrypt();
    decrypt();
}
int prime(long int pr)
{
    int i;
    j = sqrt(pr);
    for (i = 2; i <= j; i++)
    {
        if (pr % i == 0)
            return 0;
    }
    return 1;
}
void ce()
{
    int k;
    k = 0;
    for (i = 2; i < t; i++)
    {
        if (t % i == 0)
            continue;
        flag = prime(i);
```

```
        if (flag == 1 && i != p && i != q)
        {
            e[k] = i;
            flag = cd(e[k]);
            if (flag > 0)
            {
                d[k] = flag;
                k++;
            }
            if (k == 99)
                break;
        }
    }
}
long int cd(long int x)
{
    long int k = 1;
    while (1)
    {
        k = k + t;
        if (k % x == 0)
            return (k / x);
    }
}
void encrypt()
{
    long int pt, ct, key = e[0], k, len;
    i = 0;
    len = strlen(msg);
    while (i != len)
    {
```

```
        pt = m[i];
        pt = pt - 96;
        k = 1;
        for (j = 0; j < key; j++)
        {
            k = k * pt;
            k = k % n;
        }
        temp[i] = k;
        ct = k + 96;
        en[i] = ct;
        i++;
    }
    en[i] = -1;
    printf("\nTHE ENCRYPTED MESSAGE IS\n");
    for (i = 0; en[i] != -1; i++)
        printf("%c", en[i]);
}
void decrypt()
{
    long int pt, ct, key = d[0], k;
    i = 0;
    while (en[i] != -1)
    {
        ct = temp[i];
        k = 1;
        for (j = 0; j < key; j++)
        {
            k = k * ct;
            k = k % n;
        }
```

```
        pt = k + 96;
        m[i] = pt;
        i++;
    }
    m[i] = -1;
    printf("\nTHE DECRYPTED MESSAGE IS\n");
    for (i = 0; m[i] != -1; i++)
        printf("%c", m[i]);
}
```

Output

```
ENTER FIRST PRIME NUMBER
11
ENTER ANOTHER PRIME NUMBER
53
ENTER MESSAGE
Raghuveer
POSSIBLE VALUES OF e AND d ARE
3       347
7       223
17      153
19      219
23      407
29      269
31      151
37      253
41      241
43      387
47      343
59      379
61      341
```

67 163

71 271

73 57

79 79

83 307

89 409

97 193

101 381

THE ENCRYPTED MESSAGE IS

─a╖ `d·▌▌ b

THE DECRYPTED MESSAGE IS

Raghuveer

8. Write a program for congestion control using Leaky bucket algorithm.

Program

```
#include <stdio.h>
#include <stdbool.h>
#include <unistd.h>

#define BUCKET_CAPACITY 10  // Bucket capacity (in data units)
#define OUTPUT_RATE 2   // Output rate (data units per second)

typedef struct {
    int size;           // Current bucket size
    int outputRate;     // Output rate
} Bucket;

void initializeBucket(Bucket *bucket, int rate) {
    bucket->size = 0;
    bucket->outputRate = rate;
}
```

```
void leak(Bucket *bucket) {
    if (bucket->size > 0) {
        bucket->size = (bucket->size - OUTPUT_RATE) >= 0 ? (bucket->size - OUTPUT_RATE) : 0;
    }
}

bool sendData(Bucket *bucket, int dataSize) {
    if (bucket->size + dataSize <= BUCKET_CAPACITY) {
        bucket->size += dataSize;
        return true;  // Data sent successfully
    }
    return false;     // Bucket overflow (congestion)
}

int main() {
    Bucket bucket;
    initializeBucket(&bucket, OUTPUT_RATE);

    int data[] = {2, 5, 1, 4, 7, 3, 6};  // Sample data to send

    int numData = sizeof(data) / sizeof(data[0]);
    int i = 0;



    while (i < numData) {
        if (sendData(&bucket, data[i])) {
            printf("Data %d sent successfully! Bucket size: %d\n", data[i], bucket.size);
            i++;
        } else {
            printf("Bucket overflow! Waiting for bucket to empty...\n");
        }
```

```
        leak(&bucket);
        sleep(1);  // Simulate 1-second time intervals (adjust as needed)
    }

    return 0;
}
```

Output:

Data 2 sent successfully! Bucket size: 2

Data 5 sent successfully! Bucket size: 5

Data 1 sent successfully! Bucket size: 4

Data 4 sent successfully! Bucket size: 6

Bucket overflow! Waiting for bucket to empty...

Data 7 sent successfully! Bucket size: 9

Data 3 sent successfully! Bucket size: 10

Bucket overflow! Waiting for bucket to empty...

Bucket overflow! Waiting for bucket to empty...

Data 6 sent successfully! Bucket size: 10

==========

9. Write a program for frame sorting technique used in buffers.

Program

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
char str[3 * 127];
struct frame
{
    char text[3];
```

```
    int seq_no;
} fr[127], shuf_ary[127];
int assign_seq_no()
{
    int k = 0, i, j;
    for (i = 0; i < strlen(str); k++)
    {
        fr[k].seq_no = k;
        for (j = 0; j < 3 && str[i] != '\0'; j++)
            fr[k].text[j] = str[i++];
    }
    printf("\nAfter assigning sequence numbers:\n");
    for (i = 0; i < k; i++)
        printf("%d:%s ", i, fr[i].text);
    return k;
}
void generate(int *random_ary, const int limit)
{
    int r, i = 0, j;
    while (i < limit)
    {
        r = rand() % limit;
        for (j = 0; j < i; j++)
            if (random_ary[j] == r)
                break;
        if (i == j)
            random_ary[i++] = r;
    }
}
void shuffle(const int no_frames)
{
```

```
    int i, k = 0, random_ary[no_frames];
    generate(random_ary, no_frames);
    for (i = 0; i < no_frames; i++)
        shuf_ary[i] = fr[random_ary[i]];
    printf("\n\nAFTER SHUFFLING:\n");
    for (i = 0; i < no_frames; i++)
        printf("%d:%s ", shuf_ary[i].seq_no, shuf_ary[i].text);
}
void sort(const int no_frames)
{
    int i, j, flag = 1;
    struct frame hold;
    for (i = 0; i < no_frames - 1 && flag == 1; i++)
    {
        flag = 0;
        for (j = 0; j < no_frames - 1 - i; j++)
            if (shuf_ary[j].seq_no > shuf_ary[j + 1].seq_no)
            {
                hold = shuf_ary[j];
                shuf_ary[j] = shuf_ary[j + 1];
                shuf_ary[j + 1] = hold;
                flag = 1;
            }
    }
}
int main()
{
    int no_frames, i;
    printf("Enter the message: ");
    fgets(str, sizeof(str), stdin);
    no_frames = assign_seq_no();
```

```
    shuffle(no_frames);
    sort(no_frames);
    printf("\n\nAFTER SORTING\n");
    for (i = 0; i < no_frames; i++)
        printf("%s", shuf_ary[i].text);
    printf("\n\n");
}
```

Output

Enter the message: Network Programming

After assigning sequence numbers:

0:Net 1:wor 2:k P 3:rog 4:ram 5:min 6:g

AFTER SHUFFLING:

6:g 1:wor 5:min 3:rog 2:k P 0:Net 4:ram

AFTER SORTING

Network Programming