1. **BIT STUFFING**

**#include<stdio.h>**

**#include<conio.h>**

**#include<string.h>**

**void main()**

**{**

**int a[20],b[30],i,j,k,count,n;**

**printf("enter frame length:");**

**scanf("%d",&n);**

**printf("enter input frame(0's&1's only):");**

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

**scanf("%d",&a[i]);**

**i=0;count=1;j=0;**

**while(i<n)**

**{**

**if(a[i]==1)**

**{**

**b[j]=a[i];**

**for(k=i+1;a[k]==1&&k<n&&count<5;k++)**

**{**

**j++;**

**b[j]=a[k];**

**count++;**

**if(count==5)**

**{**

**j++;**

**b[j]=0;**

**}**

**i=k;**

**}**

**}**

**else**

**{**

**b[j]=a[i];**

**}**

**i++;**

**j++;**

**}**

**printf("After stuffing the frame is:");**

**for(i=0;i<j;i++)**

**printf("%d",b[i]);**

**getch();**

**}**

1. **CHARACTER STUFFING**

**#include<stdio.h>**

**#include<conio.h>**

**#include<string.h>**

**#include<process.h>**

**void main()**

**{**

**int i=0,j=0,n,pos;**

**char a[20],b[50],ch;**

**printf("enter string:\n");**

**scanf("%s",&a);**

**n=strlen(a);**

**printf("enter position\n");**

**scanf("%d",&pos);**

**if(pos>n)**

**{**

**printf("invalid position,Enter again:");**

**scanf("%d",&pos);**

**}**

**printf("enter the character\n");**

**ch=getche();**

**b[0]='d';**

**b[1]='l';**

**b[2]='e';**

**b[3]='s';**

**b[4]='t';**

**b[5]='x';**

**j=6;**

**while(i<n)**

**{**

**if(i==pos-1)**

**{**

**b[j]='d';**

**b[j+1]='l';**

**b[j+2]='e';**

**b[j+3]=ch;**

**b[j+4]='d';**

**b[j+5]='l';**

**b[j+6]='e';**

**j=j+7;**

**}**

**if(a[i]=='d'&&a[i+1]=='l'&& a[i+2]=='e')**

**{**

**b[j]='d';**

**b[j+1]='l';**

**b[j+2]='e';**

**j=j+3;**

**}**

**b[j]=a[i];**

**i++;**

**j++;**

**}**

**b[j]='d';**

**b[j+1]='l';**

**b[j+2]='e';**

**b[j+3]='e';**

**b[j+4]='t';**

**b[j+5]='x';**

**b[j+6]='\0';**

**printf("\n frame after stuffing: \n");**

**printf("%s",b);**

**getch();**

**}**

1. **CHARACTER COUNT**

**#include<stdio.h>**

**#include<conio.h>**

**#include<string.h>**

**char data[20][20];**

**int n;**

**void main()**

**{**

**int i,ch,j;**

**char tmp[20][20];**

**printf("Enter the number of frames:");**

**scanf("%d",&n);**

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

**{**

**if(i!=0)**

**{**

**printf("frame%d:",i);**

**fflush(stdin);**

**gets(data[i]);**

**}**

**}**

**/\*saving frame with count and data\*/**

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

**{**

**tmp[i][0]=49+strlen(data[i]);**

**tmp[i][1]='\0';**

**strcat(tmp[i],data[i]);**

**}**

**printf("\n\t\tAT THE SENDER:\n");**

**printf("Data as frames:\n");**

**for(i=1;i<=n;i++)**

**{**

**printf("Frame%d:",i);**

**puts(tmp[i]);**

**}**

**printf("Data transmitted:");**

**for(i=1;i<=n;i++)**

**printf("%s",tmp[i]);**

**printf("\n\t\tAT THE RECEIVER\n");**

**printf("The data received:");**

**for(i=1;i<=n;i++)**

**{**

**ch=(int)(tmp[i][0]-49);**

**for(j=1;j<=ch;j++)**

**data[i][j-1]=tmp[i][j];**

**data[i][j-1]='\0';**

**}**

**printf("\n The data after removing count char:");**

**for(i=1;i<=n;i++)**

**printf("%s",data[i]);**

**printf("\n The data in frame form:\n");**

**for(i=1;i<=n;i++)**

**{**

**printf("Frame%d:",i);**

**puts(data[i]);**

**}**

**getch();**

**}**

1. **CRC POLYNOMIALS**

**#include<stdio.h>**

**#include<conio.h>**

**int main(void)**

**{**

**int data[50],div[16],rem[16];**

**int datalen, divlen, i,j,k;**

**int ch;**

**printf("Enter the data: ");**

**i = 0;**

**while((ch = fgetc(stdin)) != '\n')**

**{**

**if(ch == '1')**

**data[i] = 1;**

**else**

**data[i] = 0;**

**i++;**

**}**

**datalen = i;**

**printf("\nEnter the divisor: ");**

**i = 0;**

**while((ch = fgetc(stdin)) != '\n')**

**{**

**if(ch == '1')**

**div[i] = 1;**

**else**

**div[i] = 0;**

**i++;**

**}**

**divlen = i;**

**for(i = datalen ; i < datalen + divlen - 1 ; i++)**

**data[i] = 0;**

**datalen = datalen + divlen - 1;**

**for(i = 0 ; i < divlen ; i++)**

**rem[i] = data[i];**

**k = divlen-1;**

**while(k < datalen)**

**if(rem[0] == 1)**

**{**

**for(i = 0 ; i < divlen ; i++)**

**rem[i] = rem[i] ^ div[i];**

**}**

**else**

**{**

**if(k == datalen-1)**

**break;**

**for(i = 0 ; i < divlen-1 ; i++)**

**{**

**rem[i] = rem[i+1];**

**printf("%d",rem[i]);**

**}**

**rem[i] = data[++k];**

**printf("%d\n",rem[i]);**

**}**

**j=1;**

**for(i = datalen - divlen + 1 ; i < datalen ; i++)**

**{**

**data[i] = rem[j++];**

**}**

**printf("\nThe data to be sent is\n");**

**for(i = 0 ; i < datalen ; i++)**

**printf("%d",data[i]);**

**getch();**

**return 0;**

**}**

1. **DISTANCE VECTOR ROUTING ALGORITHM**

**#include<stdio.h>**

**struct node**

**{**

**unsigned dist[20];**

**unsigned from[20];**

**}**

**rt[10];**

**int main()**

**{**

**int dmat[20][20];**

**int n,i,j,k,count=0;**

**printf("\nEnter the number of nodes : ");**

**scanf("%d",&n);**

**printf("\nEnter the cost matrix :\n");**

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

**for(j=0;j<n;j++)**

**{**

**scanf("%d",&dmat[i][j]);**

**dmat[i][i]=0;**

**rt[i].dist[j]=dmat[i][j];**

**rt[i].from[j]=j;**

**}**

**do**

**{**

**count=0;**

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

**for(j=0;j<n;j++)**

**for(k=0;k<n;k++)**

**if(rt[i].dist[j]>dmat[i][k]+rt[k].dist[j])**

**{**

**rt[i].dist[j]=rt[i].dist[k]+rt[k].dist[j];**

**rt[i].from[j]=k;**

**count++;**

**}**

**}**

**while(count!=0);**

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

**{**

**printf("\n\nState value for router %d is \n",i+1);**

**for(j=0;j<n;j++)**

**{**

**printf("\t\nnode %d via %d Distance%d",j+1,rt[i].from[j]+1,rt[i].dist[j]);**

**}**

**}**

**printf("\n\n");**

**}**

1. **SHORTEST PATH ROUTING ALGORITHM**

**#include<stdio.h>**

**#include<conio.h>**

**#define INFINITY 9999**

**#define MAX 10**

**void dijkstra(int G[MAX][MAX],int n,int startnode);**

**int main()**

**{**

**int G[MAX][MAX],i,j,n,u;**

**printf("Enter no. of vertices:");**

**scanf("%d",&n);**

**printf("\nEnter the adjacency matrix:\n");**

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

**for(j=0;j<n;j++)**

**scanf("%d",&G[i][j]);**

**printf("\nEnter the starting node:");**

**scanf("%d",&u);**

**dijkstra(G,n,u);**

**return 0;**

**}**

**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;**

**//pred[] stores the predecessor of each node**

**//count gives the number of nodes seen so far**

**//create the cost matrix**

**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];**

**//initialize pred[],distance[] and visited[]**

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

**{**

**distance[i]=cost[startnode][i];**

**pred[i]=startnode;**

**visited[i]=0;**

**}**

**distance[startnode]=0;**

**visited[startnode]=1;**

**count=1;**

**while(count<n-1)**

**{**

**mindistance=INFINITY;**

**//nextnode gives the node at minimum distance**

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

**if(distance[i]<mindistance&&!visited[i])**

**{**

**mindistance=distance[i];**

**nextnode=i;**

**}**

**//check if a better path exists through nextnode**

**visited[nextnode]=1;**

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

**if(!visited[i])**

**if(mindistance+cost[nextnode][i]<distance[i])**

**{**

**distance[i]=mindistance+cost[nextnode][i];**

**pred[i]=nextnode;**

**}**

**count++;**

**}**

**//print the path and distance of each node**

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

**if(i!=startnode)**

**{**

**printf("\nDistance of node%d=%d",i,distance[i]);**

**printf("\nPath=%d",i);**

**j=i;**

**do**

**{**

**j=pred[j];**

**printf("<-%d",j);**

**}**

**while(j!=startnode);**

**}**

**}**

1. **LINK STATE ROUTING ALGORITHM**

**#include <stdio.h>**

**#include <string.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++)**

**{**

**printf("\n%d->%d:",i,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("\n<--%d",last[w]);w=last[w];**

**}**

**printf("\n Shortest path cost:%d",dist[i]);**

**}}**

1. **BROADCAST ROUTING**

**#include<stdio.h>**

**int a[10][10],n;**

**main()**

**{**

**int i,j,root;**

**printf("Enter no.of nodes:");**

**scanf("%d",&n);**

**printf("Enter adjacent matrix\n");**

**for(i=1;i<=n;i++)**

**for(j=1;j<=n;j++)**

**{**

**printf("Enter connecting of %d>%d::",i,j);**

**scanf("%d",&a[i][j]);**

**}**

**printf("Enter root node:");**

**scanf("%d",&root);**

**adj(root);**

**}**

**adj(int k)**

**{**

**int i,j;**

**printf("Adjacent node of root node::\n");**

**printf("%d\n",k);**

**for(j=1;j<=n;j++)**

**{**

**if(a[k][j]==1 || a[j][k]==1)**

**printf("%d\t",j);**

**}**

**printf("\n");**

**for(i=1;i<=n;i++)**

**{**

**if((a[k][j]==0) && (a[i][k]==0) && (i!=k))**

**printf("%d",i);**

**} }**