1.BITSTUFFFING:

```
#include<iostream>
#include<string>

using namespace std;

string stuff(string data){
    string stuffed;
    int count=0;
    for(int i=0;i<data.size();i++){
        stuffed+=data[i];
        if (data[i]=='1')
        {
           count++;
        }
        if (count==5)
        {
            stuffed+='0';
            count=0;
        }


    }
    return stuffed;
}
string unstuff(string data){
    string unstuffed;
    int count=0;
    for(int i=0;i<data.size();i++){
        unstuffed+=data[i];
        if(data[i]=='1'){
            count++;
        }
        if(count==5){
            if(data[i+1]=='0'){
                i++;
            }
            count=0;
        }
        else if(data[i]=='0'){
            count=0;
        }
    }
    return unstuffed;
}
int main(){
    string data;
    cout<<"Enter data:";
    cin>>data;
    string stuffed=stuff(data);
    cout<<stuffed<<endl;
    string unstuffed=unstuff(data);
    cout<<unstuffed;
    return 0;

}
```

2.CRC:

```
def crc(data, gen_pol):
  check_val = data[:len(gen_pol)]
  for i in range(len(gen_pol), len(data) + 1):
    if check_val[0] == '1':
      check_val = bin(int(check_val, 2) ^ int(gen_pol,
2))[2:].zfill(len(gen_pol))
    if i < len(data):
      check_val = check_val[1:] + data[i]
  return check_val


#Transmitting data
data = input("Enter the data to be transmitted: ")
gen_pol = input("Enter the generating polynomial: ")
data += "0" * (len(gen_pol) - 1)
check_val = crc(data, gen_pol)
print("Check value (CRC):", check_val)
transmitted_data = data[:-(len(gen_pol) - 1)] + check_val
print("Data sent will be:", transmitted_data)


#receiving data
received_data = input("Enter the received data: ")
gen_pol = input("Enter the generating polynomial: ")
print("\n-----------------------------")
print("Data received:", received_data)
check_val = crc(received_data, gen_pol)
if '1' in check_val:
  print("\nError detected\n")
else:
  print("\nNo error detected\n")
```

3.FRAMESORT:

```python
import random as rd
msg=input("Enter the message :")
packets = [[i//3,msg[i:(i+3) if i+3 < len(msg) else None]] for i in
range(0,len(msg),3)]
print("packets :",packets)
rd.shuffle(packets)
print("shuffled packets: ",packets)
packets.sort()
print("sorted packets: ",packets)
```

4.DISTANCEVECTOR:

```python
n = int(input("Enter number of nodes : "))
print("Enter the cost matrix : ")
cost_matrix = [list(map(int, input().split())) for _ in range(n)]
rt = [[float('inf')] * (n + 1) for _ in range(n + 1)]
for i in range(1, n + 1):
    rt[i][i] = 0

for _ in range(n - 1):
    for i in range(1, n + 1):
        for j in range(1, n + 1):
            for k in range(1, n + 1):
                rt[i][j] = min(rt[i][j], rt[i][k] + cost_matrix[k-1][j-1])

for i in range(1, n + 1):
    print(f"\nRouter {chr(i + 64)}:" + "".join(f"\n\tNode {j} Distance:
{rt[i][j]}" for j in range(1, n + 1)))
```