1. Simulation of Stop and Wait Protocol

```
#include<iostream>
#include <time.h>
#include <cstdlib>
#include<ctime>
#include <unistd.h>
using namespace std;
class timer {
   private:
    unsigned long begTime;
   public:
    void start() {
     begTime = clock();
    }
 unsigned long elapsedTime() {
     return ((unsigned long) clock() - begTime) / CLOCKS_PER_SEC;
   }
  bool isTimeout(unsigned long seconds) {
     return seconds >= elapsedTime();
    }
};
int main()
{
int frames[] = {1,2,3,4,5,6,7,8,9,10};
unsigned long seconds = 5;
srand(time(NULL));
timer t;
cout<<"Sender has to send frames : ";
for(int i=0;i<10;i++)
    cout<<frames[i]<<" ";
cout<<endl;
int count = 0;
bool delay = false;
cout<<endl<<"Sender\t\t\t\tReceiver"<<endl;
do
{
    bool timeout = false;
    cout<<"Sending Frame : "<<frames[count];
    cout.flush();
```

```
    cout<<"\t\t";
    t.start();
    if(rand()%2)
    {
        int to = 24600 + rand()%(64000 - 24600)  + 1;
        for(int i=0;i<64000;i++)
            for(int j=0;j<to;j++) {}
    }
    if(t.elapsedTime() <= seconds)
    {
        cout<<"Received Frame : "<<frames[count]<<" ";
        if(delay)
        {
            cout<<"Duplicate";
            delay = false;
        }
        cout<<endl;
        count++;
    }
    else
    {
        cout<<"---"<<endl;
        cout<<"Timeout"<<endl;
        timeout = true;
    }
    t.start();
    if(rand()%2 || !timeout)
    {
        int to = 24600 + rand()%(64000 - 24600)  + 1;
        for(int i=0;i<64000;i++)
            for(int j=0;j<to;j++) {}
        if(t.elapsedTime() > seconds )
        {
            cout<<"Delayed Ack"<<endl;
            count--;
            delay = true;
        }
        else if(!timeout)
            cout<<"Acknowledgement : "<<frames[count]-1<<endl;
    }
```

```
}while(count!=10);
return 0;
}
```

Simulation of sliding window protocol for go back n ARQ!

```
#include<iostream>
#include<ctime>
#include<cstdlib>
using namespace std;
int main()
{
int nf,N;
int no_tr=0;
srand(time(NULL));
cout<<"Enter the number of frames : ";
cin>>nf;
cout<<"Enter the Window Size : ";
cin>>N;
int i=1;
while(i<=nf)
{
    int x=0;
    for(int j=i;j<i+N && j<=nf;j++)
    {
        cout<<"Sent Frame "<<j<<endl;
        no_tr++;
    }
    for(int j=i;j<i+N && j<=nf;j++)
    {
        int flag = rand()%2;
        if(!flag)
            {
                cout<<"Acknowledgment for Frame "<<j<<endl;
                x++;
            }
        else
```

```
            {    cout<<"Frame "<<j<<" Not Received"<<endl;
                 cout<<"Retransmitting Window"<<endl;
                 break;
            }
    }
    cout<<endl;
    i+=x;
}
cout<<"Total number of transmissions : "<<no_tr<<endl;
return 0;
}
```

Simulation of sliding window protocol for Selective repeat ARQ!

```
#include<bits/stdc++.h>
using namespace std;



#define TOT_FRAMES 500

#define FRAMES_SEND 10



class sel_repeat

{

   private:

   int fr_send_at_instance;

   int arr[TOT_FRAMES];
```

```
    int send[FRAMES_SEND];

    int rcvd[FRAMES_SEND];

    char rcvd_ack[FRAMES_SEND];

    int sw;

    int rw;        //tells expected frame

    public:

        void input();

        void sender(int);

        void receiver(int);

};

    void sel_repeat::input()

    {

        int n;      //no. of bits for the frame

        int m;     //no. of frames from n bits
        int i;
        cout<<"Enter the no. of bits for the sequence no. : ";

        cin>>n;

        m=pow(2,n);

        int t=0;

        fr_send_at_instance=(m/2);

        for(i=0;i<TOT_FRAMES;i++)
```

```
    {
    arr[i]=t;
    t=(t+1)%m;
    }
for(i=0;i<fr_send_at_instance;i++)
{
    send[i]=arr[i];
    rcvd[i]=arr[i];
    rcvd_ack[i]='n';
}
rw=sw=fr_send_at_instance;
sender(m);
}
void sel_repeat::sender(int m)
{
for(int i=0;i<fr_send_at_instance;i++)
{
    if(rcvd_ack[i]=='n')
    cout<<"SENDER : Frame "<<send[i]<<" is sent\n";
}
```

```
receiver(m);

}

    void sel_repeat::receiver(int m)

    {

    time_t t;

    int f;
    int j;
    int f1;

    int a1;

    char ch;

    srand((unsigned)time(&t));

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

    {

    if(rcvd_ack[i]=='n')

    {

    f=rand()%10;

    //if f=5 frame is discarded for some reason

    //else frame is correctly recieved

    if(f!=5)

    {

    for(int j=0;j<fr_send_at_instance;j++)
```

```
        if(rcvd[j]==send[i])

        {

        cout<<"reciever:Frame"<<rcvd[j]<<"recieved correctly\n";

        rcvd[j]=arr[rw];

        rw=(rw+1)%m;

        break;

        }
        int j;
        if(j==fr_send_at_instance)

        cout<<"reciever:Duplicate frame"<<send[i]<<"discarded\n";

        a1=rand()%5;

        //if a1==3 then ack is lost

        //else recieved

        if(a1==3)

        {

        cout<<"(acknowledgement "<<send[i]<<" lost)\n";

        cout<<"(sender timeouts-->Resend the frame)\n";

        rcvd_ack[i]='n';

        }

        else

        {
```

```
cout<<"(acknowledgement "<<send[i]<<" recieved)\n";

rcvd_ack[i]='p';

}

}

else

{int ld=rand()%2;

//if =0 then frame damaged

//else frame lost

if(ld==0)

{

cout<<"RECEIVER : Frame "<<send[i]<<" is damaged\n";

cout<<"RECEIVER : Negative Acknowledgement "<<send[i]<<" sent\n";

}

else

{

cout<<"RECEIVER : Frame "<<send[i]<<" is lost\n";

cout<<"(SENDER TIMEOUTS-->RESEND THE FRAME)\n";

}

rcvd_ack[i]='n';

}
```

```
}

}

for(int j=0;j<fr_send_at_instance;j++)

{

if(rcvd_ack[j]=='n')

break;

}

int i=0;

for(int k=j;k<fr_send_at_instance;k++)

{

send[i]=send[k];

if(rcvd_ack[k]=='n')

rcvd_ack[i]='n';

else

rcvd_ack[i]='p';

i++;

}

if(i!=fr_send_at_instance)

{

for(int k=i;k<fr_send_at_instance;k++)
```

```
{

send[k]=arr[sw];

sw=(sw+1)%m;

rcvd_ack[k]='n';

}

}

cout<<"Want to continue";

cin>>ch;

cout<<"\n";

if(ch=='y')

sender(m);

else

exit(0);

}

int main()

{
sel_repeat sr;
sr.input();

}
```