AIM : Simulate sliding window protocol using C++

a. Go Back N Protocol
b. Selective Repeat Protocol

PROGRAM AND OUTPUT :

```
#include <iostream>
#include <conio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
using namespace std;

#define TOT_FRAMES 500
#define FRAMES_SEND 10

class goBackN
{
private:
  int fr_send_at_instance;
  int arr[TOT_FRAMES];
  int arr1[FRAMES_SEND];
  int sw;
  int rw; // tells expected frame
public:
  goBackN();
  void input();
  void sender(int);
  void reciever(int);
};

goBackN ::goBackN()
{
  sw = 0;
  rw = 0;
}

void goBackN ::input()
{

  int n; // no of bits for the frame
  int m; // no of frames from n bits

  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 (int i = 0; i < TOT_FRAMES; i++)
  {
    arr[i] = t;
    t = (t + 1) % m;
  }

  sender(m);
}

void goBackN ::sender(int m)
{

  int j = 0;

  for (int i = sw; i < sw + fr_send_at_instance; i++)
  {
    arr1[j] = arr[i];
    j++;
  }

  for (int i = 0; i < j; i++)
    cout << " SENDER   : Frame " << arr1[i] << " is sent\n";

  reciever(m);
}

void goBackN ::reciever(int m)
{

  time_t t;
  int f;
  int f1;
  int a1;
  char ch;

  srand((unsigned)time(&t));
  f = rand() % 10;

  if (f != 5) {

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

      if (rw == arr1[i]) {

        cout << "RECIEVER : Frame " << arr1[i] << " recieved correctly\n";
        rw = (rw + 1) % m;
      }

      else
        cout << "RECIEVER : Duplicate frame " << arr1[i] << " discarded\n";
    }

    a1 = rand() % 15;
    // if a1 belongs to 0 to 3 then
    //     all ack after this (incl this one) lost
    // else
    //     all recieved

    if (a1 >= 0 && a1 <= 3) {
      cout << "(Acknowledgement " << arr1[a1] << " & all after this lost)\n";
      sw = arr1[a1];
    }

    else
      sw = (sw + fr_send_at_instance) % m;

  }

  else {

    f1 = rand() % fr_send_at_instance;

    // f1 gives index of the frame being lost

    for (int i = 0; i < f1; i++) {

      if (rw == arr1[i]) {

        cout << " RECIEVER : Frame " << arr1[i] << " recieved correctly\n";
        rw = (rw + 1) % m;
      }

      else
        cout << " RECIEVER : Duplicate frame " << arr1[i] << " discarded\n";

    }

    int ld = rand() % 2;
    // ld == 0 frame damaged
    // else frame lost
    if (ld == 0)
      cout << " RECIEVER : Frame " << arr1[f1] << " damaged\n";
    else
      cout << "            (Frame " << arr1[f1] << " lost)\n";

    for (int i = f1 + 1; i < fr_send_at_instance; i++)
      cout << " RECIEVER : Frame " << arr1[i] << " discarded\n";

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

    sw = arr1[f1];
```

```
  }
  cout << "Want to continue...";
  cin >> ch;

  if (ch == 'y')
    sender(m);
  else
    exit(0);
}

int main()
{
  goBackN gb;
  gb.input();
  getch();
}
```

OUTPUT:

```
PS C:\Users\gautam\OneDrive\Desktop> ./a
Enter the no of bits for the sequence no 4
 SENDER   : Frame 0 is sent
 SENDER   : Frame 1 is sent
 SENDER   : Frame 2 is sent
 SENDER   : Frame 3 is sent
 SENDER   : Frame 4 is sent
 SENDER   : Frame 5 is sent
 SENDER   : Frame 6 is sent
 SENDER   : Frame 7 is sent
RECIEVER : Frame 0 recieved correctly
RECIEVER : Frame 1 recieved correctly
RECIEVER : Frame 2 recieved correctly
RECIEVER : Frame 3 recieved correctly
RECIEVER : Frame 4 recieved correctly
RECIEVER : Frame 5 recieved correctly
RECIEVER : Frame 6 recieved correctly
RECIEVER : Frame 7 recieved correctly
Want to continue...n
PS C:\Users\gautam\OneDrive\Desktop>
```

SELECTIVE REPEAT PROTOCOL

```
# include <iostream>
# include <conio.h>
# include <stdlib.h>
# include <time.h>
# include <math.h>
using namespace std;

# define TOT_FRAMES 500
# define FRAMES_SEND 10
```

```
class SelectiveRepeat {
 private:
  int frameSentAtInstance;
  int arr[TOT_FRAMES];
  int send[FRAMES_SEND];
  int received[FRAMES_SEND];
  char receivedAck[FRAMES_SEND];
  int sw;
  int rw; // tells expected frame
 public:
  void input();
  void sender(int);
  void reciever(int);
};

void SelectiveRepeat :: input() {
 int n;  // no of bits for the frame
 int m;  // no of frames from n bits

 cout << "Enter the no of bits for the sequence number ";
 cin >> n;

 m = pow (2 , n);

 int t = 0;

 frameSentAtInstance = (m / 2);

 for (int i = 0 ; i < TOT_FRAMES ; i++) {
  arr[i] = t;
  t = (t + 1) % m;
 }

 for (int i = 0 ; i < frameSentAtInstance ; i++) {
  send[i] = arr[i];
  received[i] = arr[i];
  receivedAck[i] = 'n';
 }

 rw = sw = frameSentAtInstance;

 sender(m);
}

void SelectiveRepeat :: sender(int m) {
 for (int i = 0 ; i < frameSentAtInstance ; i++) {
  if ( receivedAck[i] == 'n' )
   cout << " SENDER   : Frame " << send[i] << " is sent\n";
 }
 reciever (m);

}
```

```
void SelectiveRepeat :: reciever(int m) {
 time_t t;
 int f;
 int f1;
 int a1;
 char ch;

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

 for (int i = 0 ; i < frameSentAtInstance ; i++) {
  if (receivedAck[i] == 'n') {
   f = rand() % 10;

   // if = 5 frame is discarded for some reason
   // else frame is correctly recieved

   if (f != 5) {
       int j;
    for ( j = 0 ; j < frameSentAtInstance ; j++)

     if (received[j] == send[i]) {
      cout << "RECIEVER : Frame " << received[j] << " recieved correctly\n";
      received[j] = arr[rw];
      rw = (rw + 1) % m;
      break;
     }

    if ( j == frameSentAtInstance)
     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";
            receivedAck[i] = 'n';
        }
        else {
            cout << "(Acknowledgement " << send[i] << " recieved)\n";
            receivedAck[i] = 'p';
        }
   }
   else {
    int ld = rand() % 2;

    // if = 0 then frame damaged
    // else frame lost

    if (ld == 0) {
     cout << "RECIEVER : Frame " << send[i] << " is damaged\n";
     cout << "RECIEVER : Negative acknowledgement " << send[i]  << " sent\n";
    }

    else {
     cout << "RECIEVER : Frame " << send[i] << " is lost\n";
     cout << " (SENDER TIMEOUTS --> RESEND THE FRAME)\n";
    }
    receivedAck[i] = 'n';
   }
  }
 }
int j;
 for (  j = 0 ; j < frameSentAtInstance ; j++) {
  if (receivedAck[j] == 'n')
   break;
 }
 int i = 0 ;

 for (int k = j ; k < frameSentAtInstance ; k++) {
  send[i] = send[k];

  if (receivedAck[k] == 'n')
   receivedAck[i] = 'n';
  else
   receivedAck[i] = 'p';
  i++;
 }

 if ( i != frameSentAtInstance ) {
  for ( int k = i ; k < frameSentAtInstance ; k++) {
   send[k] = arr[sw];
   sw = (sw + 1) % m;
   receivedAck[k] = 'n';
  }
 }
 cout << "Want to continue...";
 cin >> ch;
 cout << "\n";

 if (ch == 'y')
  sender(m);
 else
  exit(0);
}

int main()
{
 SelectiveRepeat sr;
 sr.input();
 getch();
}
```

OUTPUT :

```
 PS C:\Users\gautam\OneDrive\Desktop> ./a
Enter the no of bits for the sequence number 4
 SENDER   : Frame 0 is sent
 SENDER   : Frame 1 is sent
 SENDER   : Frame 2 is sent
 SENDER   : Frame 3 is sent
 SENDER   : Frame 4 is sent
 SENDER   : Frame 5 is sent
 SENDER   : Frame 6 is sent
 SENDER   : Frame 7 is sent
RECIEVER : Frame 0 recieved correctly
(Acknowledgement 0 recieved)
RECIEVER : Frame 1 recieved correctly
(Acknowledgement 1 recieved)
RECIEVER : Frame 2 recieved correctly
(Acknowledgement 2 lost)
 (SENDER TIMEOUTS --> RESEND THE FRAME)
RECIEVER : Frame 3 is lost
 (SENDER TIMEOUTS --> RESEND THE FRAME)
RECIEVER : Frame 4 recieved correctly
(Acknowledgement 4 recieved)
RECIEVER : Frame 5 recieved correctly
(Acknowledgement 5 recieved)
RECIEVER : Frame 6 recieved correctly
(Acknowledgement 6 recieved)
RECIEVER : Frame 7 recieved correctly
(Acknowledgement 7 recieved)
Want to continue...n

PS C:\Users\gautam\OneDrive\Desktop>
```

CONCLUSION : Programs to implement sliding window protocols using C++ was executed successfully,