Bhargav S

1BM18EC025

VI Sem A2 Batch

Implementation of Stop and Wait ARQ Protocol in C Language

C Code

```
#include<stdio.h>

#include <time.h>

#include <cstdlib>

#include<ctime>

#include <unistd.h>

#define RESPONSE_TIME 5


using namespace std;


// Timer class to count the time taken for receiver to respond and check if there is a timeout

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 > RESPONSE_TIME;
                }
};

// Function to display an n-bit frame
void display_frame(int arr[], int n)
{
        for(int i = 0; i < n; i++)
                printf("%d", arr[i]);
}

int main()
{
        int frames[][6] = {{0,1,0,1,1,0},{0,0,0,1,1,1},{1,1,1,0,0,0},{0,0,0,0,0,0},{1,1,1,1,1,1}}; // the
5 frames to be sent
        int Sn = 0, prev_Sn = 0;
        srand(time(NULL));
        timer t;
        printf("There are 5 frames to be sent\n");
        int count = 0;
```

```
bool delay = false;
printf("Sender\t\t\t\t\tReceiver\n");
do
{
        bool timeout = false;
        printf("Sending frame: {%d : ", count+1);
        display_frame(frames[count], 6);
        printf("}");
        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++) {}
  }
  else
        Sn = (rand()%2)?1:0;

  if(!t.isTimeout(t.elapsedTime()) && Sn != prev_Sn) //The frame is received correctly
  {
        printf("\t\tReceived frame: {%d : ", count+1);
        display_frame(frames[count], 6);
        printf("}");
        }
  else if(!t.isTimeout(t.elapsedTime()) && Sn == prev_Sn)
  {
```

```
                    printf("\t\tReceived frame is Corrupted. Resending frame.\n"); //The frame
received is corrupted
                    printf("\n");
                    prev_Sn = Sn;
                    continue;
                    }
                    else if(t.isTimeout(t.elapsedTime())) //The frame is not received
                    {
                              printf("\t\tFrame not received. Resending frame.\n");
                    printf("\n");
                    prev_Sn = Sn;
                    continue;
                    }
                    prev_Sn = Sn;
                    printf("\n");
                    count++;
          }while(count<5);


          return 0;
}
```

OUTPUT

- **Run 1**

```
There are 5 frames to be sent
          Sender                                            Receiver
Sending frame: {1 : 010110}          Frame not received. Resending frame.

Sending frame: {1 : 010110}          Frame not received. Resending frame.
Sending frame: {1 : 010110}          Frame not received. Resending frame.

Sending frame: {1 : 010110}          Received frame is Corrupted. Resending frame.

Sending frame: {1 : 010110}          Received frame is Corrupted. Resending frame.

Sending frame: {1 : 010110}          Received frame is Corrupted. Resending frame.

Sending frame: {1 : 010110}          Received frame is Corrupted. Resending frame.

Sending frame: {1 : 010110}          Received frame is Corrupted. Resending frame.

Sending frame: {1 : 010110}          Received frame is Corrupted. Resending frame.

Sending frame: {1 : 010110}          Received frame is Corrupted. Resending frame.

Sending frame: {1 : 010110}          Received frame: {1 : 010110}
Sending frame: {2 : 000111}          Received frame: {2 : 000111}
Sending frame: {3 : 111000}          Frame not received. Resending frame.

Sending frame: {3 : 111000}          Received frame is Corrupted. Resending frame.

Sending frame: {3 : 111000}          Received frame is Corrupted. Resending frame.

Sending frame: {3 : 111000}          Received frame: {3 : 111000}
Sending frame: {4 : 000000}          Received frame is Corrupted. Resending frame.

Sending frame: {4 : 000000}          Received frame is Corrupted. Resending frame.

Sending frame: {4 : 000000}          Frame not received. Resending frame.

Sending frame: {4 : 000000}          Received frame is Corrupted. Resending frame.

Sending frame: {4 : 000000}          Frame not received. Resending frame.

Sending frame: {4 : 000000}          Received frame: {4 : 000000}
Sending frame: {5 : 111111}          Received frame: {5 : 111111}

--------------------------------
Process exited after 69.39 seconds with return value 0
Press any key to continue . . .
```

- **Run 2**

```
There are 5 frames to be sent
          Sender                                    Receiver
Sending frame: {1 : 010110}          Received frame: {1 : 010110}
Sending frame: {2 : 000111}          Received frame: {2 : 000111}
Sending frame: {3 : 111000}          Received frame: {3 : 111000}
Sending frame: {4 : 000000}          Received frame is Corrupted. Resending frame.

Sending frame: {4 : 000000}          Received frame is Corrupted. Resending frame.

Sending frame: {4 : 000000}          Received frame is Corrupted. Resending frame.

Sending frame: {4 : 000000}          Frame not received. Resending frame.

Sending frame: {4 : 000000}          Received frame: {4 : 000000}
Sending frame: {5 : 111111}          Received frame: {5 : 111111}

--------------------------------
Process exited after 9.561 seconds with return value 0
Press any key to continue . . .
```

- **Run 3**

```
There are 5 frames to be sent
          Sender                                    Receiver
Sending frame: {1 : 010110}          Frame not received. Resending frame.

Sending frame: {1 : 010110}          Frame not received. Resending frame.

Sending frame: {1 : 010110}          Frame not received. Resending frame.

Sending frame: {1 : 010110}          Received frame is Corrupted. Resending frame.

Sending frame: {1 : 010110}          Received frame is Corrupted. Resending frame.

Sending frame: {1 : 010110}          Received frame is Corrupted. Resending frame.

Sending frame: {1 : 010110}          Frame not received. Resending frame.

Sending frame: {1 : 010110}          Received frame is Corrupted. Resending frame.
Sending frame: {1 : 010110}          Received frame: {1 : 010110}
Sending frame: {2 : 000111}          Received frame is Corrupted. Resending frame.

Sending frame: {2 : 000111}          Received frame is Corrupted. Resending frame.

Sending frame: {2 : 000111}          Received frame is Corrupted. Resending frame.

Sending frame: {2 : 000111}          Received frame is Corrupted. Resending frame.

Sending frame: {2 : 000111}          Received frame is Corrupted. Resending frame.

Sending frame: {2 : 000111}          Frame not received. Resending frame.

Sending frame: {2 : 000111}          Received frame: {2 : 000111}
Sending frame: {3 : 111000}          Received frame: {3 : 111000}
Sending frame: {4 : 000000}          Received frame: {4 : 000000}
Sending frame: {5 : 111111}          Received frame: {5 : 111111}

--------------------------------
Process exited after 46.26 seconds with return value 0
Press any key to continue . . .
```

Explanation

A timer class is created to visualize the timer concept.

The code covers three scenarios:

- The frame sent is received correctly. In this case, the frame stored is dumped and moved to next frame.
- The frame sent is received but corrupted. In this case, the frame is resent.
- The frame is not received and there is a timeout. In this case, the frame is resent.

The code randomizes the occurrence of these three scenarios. Thus, for run 1, 2 and 3, the outputs are different.