CS 559: Computer Systems Lab-I
Date: 12th November 2020
Assignment-9
Submission Deadline: 16th November, 2020 11:59 pm

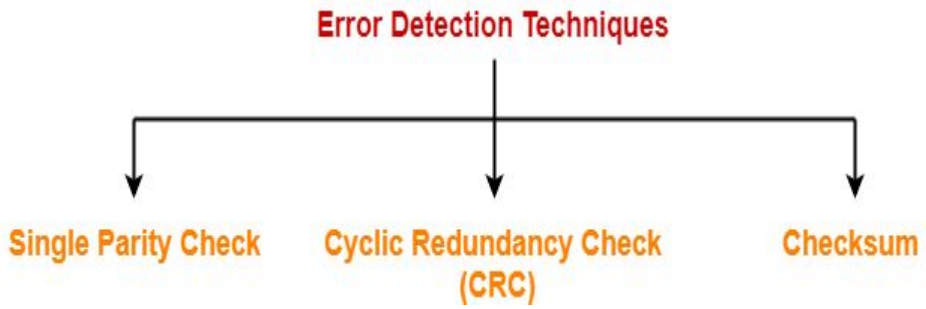
Note:

1. Store your assignments in a folder and compress it as a tar file (filename should be in this format: roll-number_assign9.tar). For example, if your roll number is 2011CS01, store your assignment as 2011CS01_assign9.tar. **Those who are unable to save as 'tar' files may opt for 'rar' files.** Also, save each program in the format given beside each question. Upload the same at the below link:

   https://www.dropbox.com/request/cgjOcOH9dOmHcq9f2ttB

2. Since we are working online, your code will be checked for similarity and you will be penalized according to the following rule:
   The similarity between 50% to less than 75%: *25% deduction*
   The similarity between 75% to 100%: *50% deduction*

3.) Avoid multiple submissions to avoid confusion during evaluation.



Checksum-

Checksum is an error detection method.
Error detection using checksum method involves the following steps-

Step-01:

At sender side,

- If m bit checksum is used, the data unit to be transmitted is divided into segments of m bits.

- The value of the checksum word is set to zero.
- All the m bit segments including the checksum are added. If carry is generated, it is appended to the sum.
- The result of the sum is then complemented using 1's complement arithmetic.
- The value so obtained is called as **checksum**, is sent along with the data

Step-2:

At receiver side,

- If m bit checksum is being used, the received data unit is divided into segments of m bits.
- All the m bit segments are added along with the checksum value. If carry is generated, it is appended to the sum.
- The value so obtained is complemented and the result is checked.

Then, following two cases are possible-

Case-01: Result = 0

If the result is zero,

- Receiver assumes that no error occurred in the data during the transmission.
- Receiver accepts the data.

Case-02: Result ≠ 0

If the result is non-zero,

- Receiver assumes that error occurred in the data during the transmission.
- Receiver discards the data and asks the sender for retransmission.

Note: For more complete explanation, refer to page 298, chapter 10: Error detection and Correction, Data Communications and Networking by Behrouz Forouzan 4th Ed.

Question 1: (Upload filename: ass9a.c)

Write a program to calculate checksum as practiced by several Internet protocols for error detection. Bear in mind that the ASCII representation of the text considers every character as one byte but 16-bits word is used for the calculation of checksum; i.e., the term 'm' in the above explanation is set to be 16. The program should accept data present in a text file. For the sake of simplicity, here, we consider that the text present in the file is 'ABCDEFGH'. Also, consider that if there cannot be two characters at the end so as not to complete the 16 bit segment, character '0' is inserted in that place. For example, if the text is 'ABC', the segments are 4142 and 4330 for the segments divided as 'AB' and 'C0' after character zero is

appended at the end. 0x41, 0x42, 0x43 and 0x30 is the hexadecimal equivalent of the ASCII representation for characters 'A', 'B', 'C' and '0' respectively.
ASCII code can be referred from the following link:

https://www.cs.cmu.edu/~pattis/15-1XX/common/handouts/ascii.html

Now, consider that the text 'ABCDEFGH' is stored in detection.txt. ASCII code for 'A', 'B', 'C', 'D', 'E', 'F', 'G' and 'H' are 65, 66, 67, …… which when represented in hex form are equivalent to 0x41, 0x42, …….. Read the content of the file in 16-bit words and calculate the checksum and represent the result in the following manner.

Sample Output:

Expected output:

At the sender side:

AB 4142
CD 4344
EF 4546
GH 4748

Checksum to send: EEEA

At the receiver side:

4142+4344+4546+4748+EEEA= FFFF
New checksum= 0000

Explanation:

'ABCDEFGH' after being divided into 16 bit segments as 'AB', 'CD', 'EF' and 'GH' have their hex equivalents as 0x4142, 0x4344, 0x4646 and 0x4748 respectively. Adding them produce 0x11114 which is one more than the size of the segment allowed (here, 16 bits). Hence, LSB is removed and added to the sum to yield final sum as 0x1115. 1's complement of this sum is 0xEEEA. This checksum along with data chunks is sent to the receiver.

At the receiver side, adding 0x4142, 0x4344, 0x4646 0x4748 and 0xEEEA produce 0x1FFFE. Again, LSB is removed and appended to the sum to produce 0xFFFF which is complemented to produce 0x0000. Hence, received data is error-free. For any value other than 0x0000, data is corrupted in the transmission path and needs to be resent by the sender.

Question 2: To write a program to perform simulation on Selective Repeat ARQ protocol. Your program should follow the algorithm given below. **(Upload filename: ass9b.c)**

ALGORITHM:

Step 1: Start the program.

Step 2: Generate a random that gives the total number of frames to be transmitted. (in the range 40 to 100)

Step 3: Set the size of the window as 8.

Step 4: Generate a random number less than or equal to the size of the current window and identify the number of frames to be transmitted at a given time.

Step 5: Transmit the frames and receive the acknowledgement for the frames sent.

Step 6: Find the remaining frames to be sent.

Step 7: Find the current window size.

Step 8: If an acknowledgement is not received for a particular frame retransmit that frame alone again.

Step 9: Repeat the steps 4 to 8 till the number of remaining frames to be send becomes zero.

Step 10: Stop the program

Sample Output

Number of frames: 55

Sending frame 1

Sending frame 2

Sending frame 3

Sending frame 4

No Acknowledgement for the frame 2

Retransmitting frame 2

Sending frame 5

Sending frame 6

No Acknowledgement for the frame 2

Retransmitting frame 2

Sending frame 3

Sending frame 4

No Acknowledgement for the frame 4

.

.

.

Sending frame 54

Sending frame 55

End of sliding window protocol

Question 3: (Upload filename: ass9c.c)

Write a C program to implement Banker's algorithm. Use the same data structures as defined in class.

Constraints:
Number of resources: 3 to 8
Number of processes: 3 to 6

Sample Input and Output

```
Enter the number of resources: 4

Enter the number of processes: 5

Enter Claim Vector: 8 5 9 7

Enter Allocated Resource Table: 2 0 1 1 0 1 2 1 4 0 0 3 0 2 1 0 1 0 3 0

Enter Maximum Claim table: 3 2 1 4 0 2 5 2 5 1 0 5 1 5 3 0 3 0 3 3

The Claim Vector is: 8 5 9 7

The Allocated Resource Table:
        2       0       1       1
        0       1       2       1
        4       0       0       3
        0       2       1       0
        1       0       3       0

The Maximum Claim Table:
```

```
        3       2       1       4
        0       2       5       2
        5       1       0       5
        1       5       3       0
        3       0       3       3

Allocated resources: 7 3 7 5
Available resources: 1 2 2 2

Process3 is executing.

The process is in safe state.
Available vector: 5 2 2 5
Process1 is executing.

The process is in safe state.
Available vector: 7 2 3 6
Process2 is executing.

The process is in safe state.
Available vector: 7 3 5 7
Process4 is executing.

The process is in safe state.
Available vector: 7 5 6 7
Process5 is executing.

The process is in safe state.
Available vector: 8 5 9 7
```