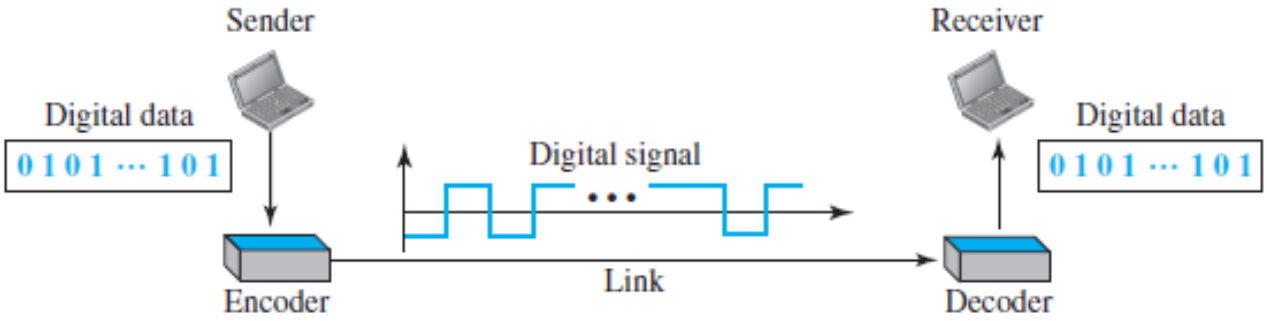
BLG337E: Principles of Computer Communications

Homework #1

Due Date: 18.10.2016, 23:00

Write a program to simulate line coding as shown in the figure below which is taken from "B. A. Forouzan, Data Communications and Networking, 5E". At the sender side, line encoder converts **digital data** (i.e., a sequence of bits) into a **digital signal**. At the receiver side, line decoder converts the **digital signal** back into the **digital data**. You can assume that the link is error-free and there is no attenuation.



The program will be executed as follows:

Program.exe sender_file_name receiver_file_name line_code_scheme

sender_file_name is the name of the file which contains the sender side digital data as a sequence of '0' and '1' characters (e.g., 0110101 … 000111). Your program will read this file and convert the data into a digital signal by using a line coding scheme whose name is given as the last command line parameter and explained below.

The link will be represented by a file whose name is constructed from your student id and the line coding scheme used (e.g., if your student id is 12345 and the line coding scheme is NRZ_L, then the link file name is "**12345_NRZ_L.txt**").

The sender writes the digital signal into this file as follows. Positive voltage level will be represented by the letter **P**, negative voltage level will be represented by the letter **N**, and zero voltage level will be represented by the letter **Z**. The signal representation of each bit will be delimited by the " | " character sequence (see the example below).

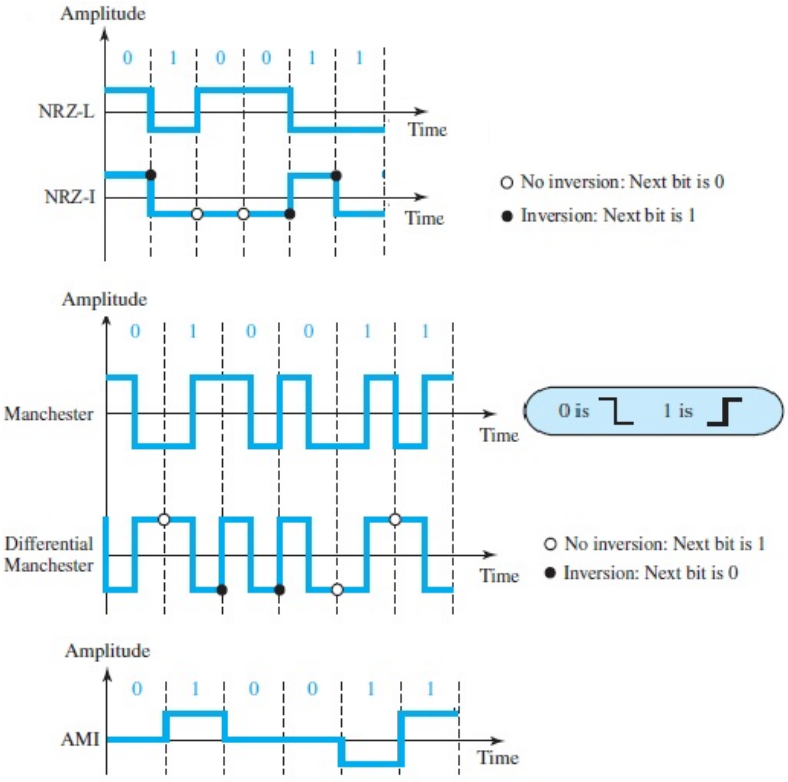
receiver_file_name is the name of the file which contains the receiver side digital data as a sequence of '0' and '1' characters (e.g., 0110101 … 000111). Your program must read the file which represents the link (e.g., **12345_NRZ_L.txt**) and convert the signals back into digital data. Then, it must write the digital data into the receiver side data file.

line_code_scheme is one of the following “NRZ_L”, “NRZ_I”, “MANCH”, “D_MANCH”, “AMI”:

- NRZ_L: **N**on-**R**eturn-to-**Z**ero scheme in which voltage **L**evel determines the value of the bit.
- NRZ_I: **N**on-**R**eturn-to-**Z**ero scheme in which change (i.e., **I**nversion) or lack of change in the voltage level determines the value of the bit.
- MANCH: **Manch**ester scheme which uses first and second halves of the signal to represent a bit.
- D_MANCH: **D**ifferential **Manch**ester scheme.
- AMI: **A**lternate **M**ark **I**nversion scheme.

Example: Assume that the sender file contains “010011”. Firstly, your **link_encoder** reads this file, character-by-character and create the **link file** whose contents must be as follows:

- For NRZ_L, the link file contains “| P | N | P | P | N | N |”
- For NRZ_I, the link file contains “| P | N | N | N | P | N |”
- For MANCH, the link file contains “| PN | NP | PN | PN | NP | NP |”
- For D_MANCH, the link file contains “| NP | PN | PN | PN | NP | PN |”
- For AMI, the link file contains “| Z | P | Z | Z | N | P |”



Then, your **link_decoder** reads the **link file** and interprets the letters P, N, and Z as positive, negative, and zero voltage levels, respectively. Then, based on these signals it constructs the receiver digital data as ‘0’ and ‘1’ characters. It writes these data into the receiver file. Beware that, sender and receiver files must have the same content at the end of this conversion.

You have to submit:

1. The **source file** which is written in **c** programming language and named as **“student_id.c**”. The source file **MUST BE** compilable and runnable in **Linux**. Please write your development environment at the start of the source file as well as your name, surname and student ID, of course within comments!

2. **A PDF report** which is named as “**student_id.pdf**”.