מטלת תכנות 1 – Noisy Channel

1. <u>מגישים-</u> איתמר שור 315129551 ועומר גיל 211453907

2. <u>הוראות שימוש-</u> הגדרנו את סדר הארגומנטים לכל אחד מהרכיבים לפי הסדר שהופיע בפרטי המטלה (תואם לדוגמאות הרצת הקוד שניתנו). לצורך בהירות נציין את הסדר:
   - ./receiver.exe <receiver port> <output file>
   - ./channel.exe <channel port> <receiver IP> <receiver port> <error prob.> <seed>
   - ./sender.exe <channel IP> <channel port> <input file>

3. <u>מבנה כללי-</u> כל רכיב תקשורת ממומש בפרויקט VS נפרד.
פונקציית המain בכל רכיב מחלוקת לשלושה שלבים, שמבוצעים בשלוש הפונקציות הבאות, בהתאמה:
   - **Init** – פונקציה המאתחלת ובונה את כל המשתנים הנדרשים (למשל יוצרת socket ומבצעת bind, פותחת קובץ קלט וכדומה) לקראת השלב הבא.
   - **Run –** פונקציה הממשת את הפונקציונליות של כל רכיב.
   - **Ready2End –** הפונקציה המשלימה של Init, סוגרת את המשתנים שinit אתחל, לפי הצורך.

תיאור הflow של שלושת הרכיבים:

- Sender:
  1. קרא 1100 בתים מהקובץ, אם קיימים. אחרת עבור ל5.
  2. מתוך הבתים שנקראו, קודד כל 11 ביט (בעזרת קוד hamming) ל15 ביט.
  3. שלח לערוץ פקטה המכילה את 1500 הבתים המקודדים.
  4. חזור ל1.
  5. אם נותרו בקובץ בתים לקריאה (כמות הקטנה מ1100 בתים), פעל באופן דומה לשלבים 2 ו3, תוך שינוי גודל הפקטה כך שיתאים למספר הבתים שנותרו.
  6. המתן לקבלת feedback מהערוץ.
  7. סיים.
- Channel:
  1. המתן לקבלת הודעה.
  2. בעת קבלת הודעה, זהה את המקור שלה (לפי מספר הport)
     אם המקור הוא השולח, עבור ל3. אחרת, המקור הוא המקבל- עבור ל6.
  3. הפוך ביטים בהודעה שהתקבלה לפי ההסתברות הנתונה כקלט.
  4. שלח את ההודעה, לאחר הרעש, אל המקבל.
  5. עבור ל1.
  6. העבר את ההודעה כפי שהיא אל השולח.
  7. סיים.
- Receiver:
  1. המתן לקבלת הודעה מהערוץ, או לקבלת End מהמשתמש.
     במקרה של הודעה מהערוץ, עבור ל2. אחרת, עבור ל5.
  2. מתוך הבתים שהתקבלו בפקטה, פענח (כולל תיקון, לפי הצורך) כל 15 ביט (בעזרת קוד hamming) ל11 ביט.
  3. כתוב את הבתים המפוענחים לקובץ הפלט.
  4. חזור ל1.
  5. סיים.

הערות מימוש:

- לכל רכיב הוספנו שני קבצים – utils.h, utils.c, המכילים מגוון פעולות עזר כמו sendall (שנלקחה מהתרגול).
- לצורך עבודה בגרנולריות של ביטים (עבור קידוד ופיענוח הודעות) יצרנו פונקציה הנקראת place_bits.
  הפונקציה מקבלת *שני מצביעים*- למקור ויעד, *מיקומים* מתאימים בכל אחד מהם, ו*כמות* ביטים רצויה.
  הפונקציה מעתיקה את כמות הביטים הרצויה מהמקור (במיקום הנתון) אל היעד (במיקום הנתון).
- בחרנו לממש את ההאזנה לstdin בreceiver באמצעות thread. הthread מחכה לקלט מהstdin (פעולה חוסמת עבורו בלבד) ובמידה ומתקבלת המחרוזת "End", מעדכן משתנה גלובלי (stop) שמתריע לחוט הראשי שעליו לסיים את הריצה.
  כדי להימנע ממצבי ביניים לא מוגדרים, השתמשנו במנעול (mutex) כדי לוודא שהמקבל, במידה וקיבל End במהלך עיבוד הודעה, יסיים את עיבודה ורק אז יעצור.
- כדי להפוך ביט בהסתברות נתונה, השתמשנו בפונקציה מתוך
  https://stackoverflow.com/questions/16255812/random-function-which-generates-1-or-0-with-given-probability

4. <u>מגבלות מימוש:</u>

נציין שאלו מגבלות מימוש הנובעות מההנחות וההוראות שהוצגו במטלה.

- השולח ממתין לקבלת feedback רק לאחר שסיים לקרוא ולהעביר לערוץ את כל המידע מהקובץ.
  לפיכך, במידה והמקבל קיבל end לפני שהשולח סיים לעבד את כל תוכן הקובץ, נקבל מצב לא מוגדר ויתכן שהשולח ימתין לfeedback לנצח.
- הנחנו כי קיים מספר שלם של הודעות בקובץ הקלט (מספר הביטים בקובץ הוא כפולה של 11). הנחה זו באה לידי ביטוי בשליחת הפקטה האחרונה (במידה וגודלה קטן מ1100 בתים) שגודלה מעוגל למספר ההודעות שלם (נתעלם מהביטים הנותרים ולא נשלח אותם).