תרגיל בית 6 – פרוייקט ב-++C

בתרגיל זה אנו נבנה סימולטור של כרטיס רשת (NIC – Network Interface Card) אשר מקבל מידע מהרשת, מסנן ומעדכן אותו לפי סט מוגדר של חוקים ואילוצים. לצורך מימוש הסימולטור נשתמש ב-Cpp בלבד תוך שימוש בעקרונות הירושה והפולימורפיזם.

בתרגיל זה נשים דגש על שני דברים:

1. נכונות – האם מסננים, מעדכנים ומעבירים מידע כפי שנדרש.
2. ניהול משאבים נכון.

יש לקרוא את הוראות התרגיל עד הסוף, ורק אח"כ להתחיל לעבוד. כמו כן מומלץ לחלק עבודה.
שימו לב: קובץ ההוראות אמנם ארוך, אבל הפתרון די קצר. עם זאת, אל תחכו לרגע האחרון.

חלק א' – קצת על רשתות תקשורת

האינטרנט שלנו עובד בעזרת שליחת וקבלת מידע בינארי ממחשבים ורכיבי חומרה שונים המחוברים זה לזה. המידע עצמו לא נשלח באופן רציף, אלא הוא מחולק לחבילות קטנות (packets) שנשלחות באופן עצמאי. באופן כללי כל חבילה יכולה לשקול בטווח של 32 בתים עד מספר קילו בתים, אך בתרגיל זה נניח שיש שלושה גדלים אפשריים בלבד.

מושגים בסיסיים:

כתובת MAC (Medioum Access Control): כתובת MAC היא מזהה ייחודי שמוקצה לכל NIC ברמת החומרה. היא מיועדת לשימוש ברשתות מקומיות, והיא מאפשרת לזהות באופן חד-משמעי את התחנה (המחשב) בתוך הרשת המקומית.

כתובת MAC היא רצף של **6 בתים (48 ביט)**, והיא מיוצגת בדרך כלל בפורמט הקסדצימלי,
למשל: FC:4D:D4:12:34:56
שזה למעשה 6 ערכים בין 00 ל- FF, מופרדים בנקודתיים.

בתוך המחשב, נוכל לייצג את כתובת ה- MAC כמערך של שישה בתים או כ- unsigned long,
למשל עבור הדוגמה לעיל, ניתן לייצג כתובת MAC באופנים הבאים:

```
unsigned long mac = (0xFC << 40) |
                    (0x4D << 32) |
                    (0xD4 << 24) |
                    (0x12 << 16) |
                    (0x34 << 8)  |
                    (0x56);
char mac[6] = {0xFc, 0x4D, 0xD4, 0x12, 0x34, 0x56};
```

בכל חבילה שנשלחת ברשת קיימות שתי כתובות MAC – כתובת השולח וכתובת היעד, לכן כאשר מכשיר מקבל חבילת רשת אשר **אינה** ממוענת אליו הוא "זורק" אותה, כלומר מוחק ולא מתייחס.

כתובת IP: זהו Unsigned Integer (4 בתים) אשר מציין איזה מחשב שלח את החבילה, או לאן היא ממוענת. כשאנחנו כותבים כתובת IP, אנו עושים זאת בפורמט הבאה: כל בית ב-integer מיוצג בצורה דצימלית(0-255), ובין כל שני בתים יש תו 'נקודה'. למשל, כתובת ה-IP הזו:
4.52.133.12 מייצגת את המספר הבא:

```
unsigned int ip = (4 << 24) |
                  (52 << 16)|
                  (133 << 8)|
                  (12);
char ip[6] = {4, 52, 133, 12};
```

מסכה (Mask): ניתן לייצג **קבוצה של מספר כתובות IP** ע"י **ציון מספר הביטים** (משמאל לימין) אליהם יש להתייחס ,בעוד כל שאר הביטים הם Don't care. ציון מספר הביטים נעשה ע"י הוספת סלאש (/) לאחר כתובת ה-IP, כתיבת המספר (בין 0 ל-32). לדוגמה, המסכה הבאה: 255.63.1.1/15 מציינת כי יש להתייחס ל- 15 הביטים השמאליים ביותר (MSB) בכתובת ה-IP, ולשאר ה-17 בתור Don't care.

<u>הבהרה:</u> אתם בטח שואלים את עצמכם מה ההבדל בין כתובת ה-MAC לכתובת ה-IP. כאשר מידע עובר ברשת, הוא עשוי לעבור דרך מספר תחנות ביניים, **כתובות IP** מזהות את השולח המקורי ואת היעד הסופי של החבילה – כלומר, מאיפה היא נשלחה ולאן היא אמורה להגיע. לעומת זאת, **כתובת MAC** משמשת ברמת התחנה המקומית: בכל קפיצה (hop) בדרך, כתובות ה- MAC מתארות את התחנה הנוכחית שמעבירה את החבילה, ואת התחנה שאליה היא נשלחת בשלב הבא.

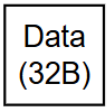
כלומר, כתובות ה- IP בחבילה נשארות קבועות לאורך כל הדרך, בעוד שכתובות ה- MAC מתעדכנות בכל תחנה – בהתאם לשולח והמקבל המקומיים.

פורט (port): זהו Short (2 בתים) אשר מציין מספר אפליקציה בתוך המחשב (לכל אפליקציה המחשב מקצה מספר ייחודי). לא ניכנס למשמעות מעבר לכך, רק נגיד שפורט מיוצג בצורה דצימלית רגילה, ונמצא בטווח 0-65535.

<u>מהי פקטה?</u>

פקטה (Packet) היא יחידת מידע קטנה שמועברת ברשת מחשבים. במקום לשלוח קובץ שלם כמקשה אחת, המידע מחולק לפקטות קטנות – כל אחת עטופה במידע נוסף שמאפשר לנתב אותה נכון, לוודא את תקינותה, ולבסוף להרכיב מחדש את התוכן בצד המקבל. כל פקטה בנויה כשכבות, כאשר כל שכבה עוטפת את המידע של השכבה הקודמת ומוסיפה מידע חיוני לצורך הניתוב, ההגנה או הזיהוי. בתרגיל זה, הפקטות מורכבות מארבע שכבות:

1. שכבת היישום - L5
   - גודל: 32 בתים
   - תוכן: **המידע עצמו** – התוכן שהיישום רוצה לשלוח, או אמור לקבל.
   - **שימו לב:** גודל זה קבוע בתרגיל.



2. שכבת התעבורה - L4
מערכת ההפעלה מכינה שכבה זו. כאשר האפליקציה רוצה להעביר מידע בגודל כלשהו, היא מעבירה אותו למערכת ההפעלה אשר מחלקת אותו לחתיכות קטנות (בתרגיל זה 32 בתים) ומוסיפה את פורט המקור (המספר המזהה של האפליקציה השולחת) ופורט היעד (המספר המזהה של האפליקציה המקבלת במכשיר היעד) ומעבירה לטיפול כרטיס הרשת.
   - גודל: 8 בתים
   - תוכן:
     - 2 בתים – פורט מקור (מי השולח)
     - 2 בתים – פורט יעד (למי מיועד)
     - 4 בתים – כתובת, מייצגת את המקום בזכרון אליו המידע אמור להיכתב.
   - מבנה:

3. שכבת הרשת - L3
כרטיס הרשת מכין שכבה זו. כרטיס הרשת מקבל פקטה מסוג L4 ממערכת ההפעלה ובהתאם למידע שיש ברשותו מכין את המידע למעבר ברשת.

- גודל: 16 בתים
- תוכן:
  - 4 בתים – כתובת IP של המקור.
  - 4 בתים – כתובת IP של היעד.
  - 4 בתים – זמן חיים (Time to Live – TTL), זהו מונה שמתחיל עם מספר כלשהו כאשר הפקטה נוצרת וכל מכשיר ביניים שמעביר אותה מחסיר ממנו 1, כאשר המונה מגיע ל-0 הפקטה נזרקת. תפקיד שדה זה חשוב מאוד שכן מונע מפקטה לנוע לנצח ברשת.
  - 4 בתים – Checksum – סכום הבתים של שכבות L5,L4,L3 – למעט שדה זה. מטרתו לבדוק את תקינות הנתונים.
- מבנה:

src ip	dst ip	time to live	cs	src port	dst port	Addrs	Data (32B)

4. שכבת ה-MAC – L2
גם את שכבה זו כרטיס הרשת מכין, אך היא מיועדת למעבר ברשת המקומית. כאשר הפקטה מוכנה היא יכולה לצאת לרשת.

- גודל: 16 בתים
- מבנה:
  - 6 בתים: כתובת MAC של המקור(כרטיס הרשת היוצר)
  - 6 בתים – כתובת MAC של היעד (כרטיס הרשת הבא שאליו החבילה עוברת)
  - 4 בתים – Checksum – סכום הספרות של כל הפקטה, למעט השדה הזה, לצורך בדיקת תקינות כללית.
- מבנה:

src mac	dst mac	src ip	dst ip	time to live	cs	src port	dst port	Addrs	Data (32B)	cs

חלק ב' – קצת על כרטיס רשת

לכרטיס הרשת ישנם תפקידים רבים, בתרגיל זה אנו נתעסק באחד מהם והוא לחבר בין רשת לוקאלית (לדוגמה ה-wi-fi בו אתם משתמשים) לעולם החיצוני. לכל כרטיס רשת ישנה כתובת IP, ומספר מסיכה (mask), אשר ביחד מתארים את מרחב הכתובות ברשת הלוקאלית של הכרטיס. כל הודעה אשר ממוענת לכתובת השייכת למרחב תכנס ע"י ה-NIC, אשר יבדוק שאין שגיאות, יעדכן את הפקטה ויעבירה ליעד הבא. את כל הפקטות שנכנסות לרשת, ה-NIC כותב לתור אשר נקרא Receive Queue (RQ), ואליו מחובר רכיב חומרתי ייעודי שמוציא את הפקטה הלאה. עבור פקטות שיוצאות מהרשת, או רק עוברות דרך הכרטיס ברשת ה-NIC יבדוק ויעדכן, ולבסוף יעברן לתור אחר אשר נקרא Transmit Queue (TQ). בנוסף ה-NIC מריץ גם אפליקציות, ולכן חלק מן הפקטות יכולות להיות מיועדות אליו, במקרה זה ה-NIC בודק שאכן התקשורת קיימת (יתואר בהמשך), במידה וכן ישמור את המידע שקיבל (L5) במקום מוגדר בזכרון.

הבהרה: בתרגיל זה לא נממש את התורים, או שימוש בזכרון שמיועד לכרטיס הרשת, אלא רק נדפיס את התוכן שאמור לעבור ליעדים הרלוונטיים.

חלק ג' – ה-NIC שלנו

הסימולטור (NIC) שנבנה בתרגיל זה הוא תוכנה אשר תקבל כקלט שמות של שני קבצים. הראשון יתאר את פרמטרי הכרטיס, והשני יכיל פקטות מסוגים L3,L4,L5 שכרטיס הרשת אמור לעבד ולהעביר. לכרטיס הרשת מוגדרים שלושה מרחבי זכרון:

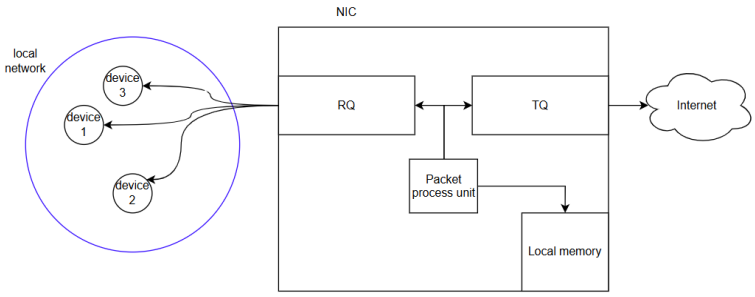
- Receive Queue (RQ) - תור שאליו נשלחות כל הפקטות שמיועדות לרשת הלוקאלית.
  בתרגיל זה התור יתואר כווקטור של מחרוזות (string), אשר יכתבו אליו כלל הפקטות הנכנסות.
- Transmit Queue (TQ) - תור שאליו משלחות כל הפקטות שלא מיועדות לרשת הלוקאלית. גם תור זה יתואר כווקטור של מחרוזות (string).
- Local Memory - נמצא בזכרון של הסימולטור, עבור כל תקשורת פתוחה, אשר מתוארת על ידי פורט מקור, ופורט יעד יוגדר מערך של בתים (char[]) בו ישמר המידע. עבור כל תקשורת פתוחה, הסימולטור ישמור את המבנה הבא:

```
struct open_port {
   short dest_port;
   short source_port;
   char data[64];
}
```

<u>הערה</u>: בתרגיל זה לא נציין כיצד הפקטות מתקבלות בכרטיס, אלא רק לאן מועברות.

בסוף התרגיל תדפיסו את תוכן שלושת מרחבי הזכרון.

להלן תרשים להמחשה:



עצרו! אם משהו לא מובן עד כאן זה הזמן לשאול שאלות בפורום ובשעות הקבלה. כאן סיימנו עם ההקדמה ונתחיל בתרגיל. **בתרגיל זה אסור להשתמש בספריית sstream.**

חלק ד' – מימוש ממשק לעיבוד הפקטה

כפי שבוודאי שמתם לב, כרטיס הרשת אמור להצליח לעבד פקטות מסוגים שונים, ולקבל החלטות בנוגע לכתיבה שלהם בזכרון או השמטתן. על מנת למנוע שכפול קוד, ואפשרויות צמיחה נרצה לממש את התהליך הנ"ל באופן גנרי. ולכן נממש את המחלקה generic_packet אשר מכילה 3 מתודות וירטואליות טהורות:

- Validate_packet: מוודא שהפקטה תקינה (יתואר בהמשך).
- Proccess_packet: יעדכן את הפקטה (האובייקט עצמו) ויחזיר את מרחב הזכרון אליו אמורה להכתב.
- As_string: מחזיר מחרוזת שתתאר את הפקטה.

את האובייקט וכלל חתימות הפונקציות המלאות עם תיעוד ניתן למצוא בקובץ packets.hpp/

כעת נממש שלוש מחלקות אשר יירשו מ- generic_packet ויטפלו בסוגים שונים של פקטה.

1. פקטות מסוג L4 (l4_packet):

- פקטות אלה מיועדות לכרטיס הרשת עצמו. יש לבדוק שאכן קיימת תקשורת בין פורט המקור לפורט היעד, במקרה וכן יש לכתוב את המידע החל מאינדקס address של המערך הרלוונטי. במידה ואין ערוץ פתוח לא צריך לעשות כלום (לזרוק את הפקטה). במידה והאינדקס אינו חוקי (יגרור שגיאת זכרון) יש להשמיט את הפקטה.
- פקטות מסוג L4 יתוארו כמחרוזת כאשר פורט המקור יופיע ראשון, אחריו בהפרדה של '|' יופיע פורט היעד ואחריו עם הפרדה של '|' יופיע אינדקס היעד ולבסוף יופיע כל המידע בפורמט הקסדצימלי עם רווח בין כל בית.
לדוגמה עבור פורט מקור 2500, פורט יעד 2000, כתובת 0 ומידע שמכיל בית ראשון 0xDD ובשאר אפסים הפקטה תראה כך: 2500|2000|0|DD 00 00 00 00 00 ... 00

2. פקטות מסוג L3 (l3_packet):

נחלק לחמישה מקרים – **בכל המקרים, קודם יש** לבצע בדיקת checksum ו-TTL חיובי ממש – פקטה לא תקינה תזרק (אין צורך לבצע פעולות נוספות):

2.1 <u>פקטה נכנסת לרשת</u> – כאשר כתובת המקור **לא** שייכת לרשת הפנימית וכתובת היעד שייכת לרשת הפנימית של כרטיס הרשת - כרטיס הרשת יפחית 1 ממונה ה-TTL (אם כעת TTL=0 יש לזרוק את הפקטה), יעדכן את שדה הchecksum ויכתוב את הפקטה החדשה כמחרוזת בתור RQ.

2.2. <u>פקטה יוצאת מהרשת</u> – כאשר כתובת היעד **לא** שייכת לרשת הפנימית וכתובת המקור כן – יבוצעו כלל הבדיקות שתוארו, כתובת השולח תוחלף בכתובת כרטיס הרשת, מונה ה-TTL יופחת ב-1 (אם כעת TTL=0 יש לזרוק את הפקטה), יעדכן את שדה הchecksum ויכתוב את הפקטה כמחרוזת בתור TQ.

2.3. <u>פקטה ש"רק" עוברת</u> – כתובת היעד וכתובת המקור לא שייכות לרשת הפנימית - על הסימולטור לבצע בדיוק את אותן הפעולות כמו פקטה ש**נכנסת** לרשת, אך לבסוף לכתוב את הפקטה בתור **TQ**.

2.4. <u>פקטה שמיועדת לכרטיס הרשת עצמו</u> – כלומר כתובת יעד ה-IP זהה לכתובת ה-NIC.

- ראשית הכרטיס יבצע בדיקת checksum ו-TTL, פקטה לא תקינה תזרק.
- כרטיס הרשת יסיר את שכבה 3, ויטפל בפקטת L4 שנשארה כפי שתואר.

2.5. <u>פקטה שכתובת המקור וכתובת היעד שייכות לרשת פנימית</u> – יש לזרוק אותה.

- פקטות מסוג L3 יתוארו כמחרוזת, כאשר תופיע כתובת IP של השולח, לאחר מכן של היעד, לאחר מכן ערך ה TTL ולבסוף checksum. דוגמה לפקטה כנ"ל היא:
4.52.123.8|4.52.123.6|23|1036|2500|2000|0|DD 00 ... 00
בדוגמה זו כתובת השולח היא 4.52.123.8, ערך ה-TTL הוא 23 וערך הcs הוא סכום כל **הבתים**: למשל את המספר 2000 כותבים כ -0x07D0, סכום הבתים: 7+208=215. בסך הכל עבור פקטה זו:

$$cs = 4 + 52 + 123 + 8 + 4 + 52 + 123 + 6 + 23 + 9 + 196 + 7 + 208 + 0 + 221 + 0 + .. + 0 = 1036$$

3. פקטות מסוג L2 (l2_packet):

- יש לבדוק שכתובת MAC היעד הוא כרטיס הרשת, במידה ולא יש לזרוק את הפקטה.
- יש לבדוק תקינות פקטה ע"י וידוא שערך ה-checksum של שכבה זו שווה לסכום כלל הבתים בפקטה (ללא שדה זה).
- להסיר את שכבה L2 ולטפל בשכבה L3 שנשארה כפי שתואר קודם.
- כתובת MAC נכתבת בבסיס הקסדצימלי, ובתים מופרדים ע"י ':'.
למשל A1:12:57:9F:00:01.

- בשכבה זו כתובת השולח תכתב ראשונה, אחריה כתובת היעד. **בסוף** הפקטה יופיע שדה הchecksum.
- דוגמה לפקטה שלמה:

A1:12:57:9F:00:01| C2:02:01:90:10:02|4.52.123.8|4.52.123.6|23|1036|2500|2000|0|DD 00 … 00|**1963**

רמז: תשאלו את עצמכם מה משותף לכל הפקטות, ואיך ניתן לחסוך בעבודה וזמן ע"י תכנון נכון.

שימו לב – במחלקה הגנרית generic_packet קיימת פונקצייה נוספת **עם מימוש**, תקראו את התיעוד שלה ותחשבו איך פונקציה זו יכולה להקל עליכם במימוש.

את הקוד יש לכתוב בקבצים L2.cpp,L3.cpp,L4.cpp,L2.h,L3.h,L4.h

עצרו! אם יש משהו שלא ברור בנוגע לאופן פעולת המחלקות generic_packet, l2_packet, l3_packet, l4_packet – זה הזמן לשאול שאלות בפורום.

חלק ה' – שלב ראשון בבניית הסימולטור, נייצר מפעל

כעת החלק הקשה מאחורינו. משום שכרטיס הרשת יכול לקבל כקלט הרבה סוגים של פקטות, נממש פונקציית מפעל לכרטיס. מה זה מפעל?

שיטת **המפעל (Factory Method)** בתכנות היא תבנית עיצוב (Design Pattern) נפוצה מאוד, במיוחד בעולם של תכנות מונחה עצמים (OOP). המטרה של השיטה היא **לאפשר יצירה של אובייקטים מבלי לחשוף את הלוגיקה הפנימית של יצירתם**. במקום ליצור אובייקטים ישירות עם new, אנחנו מגדירים **"מפעל" (factory)** שידע ליצור עבורנו את האובייקט הרצוי, בהתאם לצורך.

יתרונות מרכזיים לתבנית עיצוב זו הם הפרדה בין יצירה לשימוש, גמישות, וניתנת להרחבה בקלות.

בתרגיל זה, המפעל יקבל מחרוזת שתתאר פקטה מסוג כלשהו, בהתאם לסוג הפקטה המפעל ייצר את האובייקט הרצוי ויחזיר מצביע למחלקה הגנרית generic_packet. ניתן למצוא את חתימת המפעל בקובץ nic_sim.hpp.

מומלץ ואף רצוי להוסיף שדות או פונקציות, אך אסור להשנות או למחוק את מה שקיים.

חלק ו' – סיום בניית הסימולטור

הסימולטור יקבל כארגומנט שמות של 2 קבצים. הראשון הוא קובץ פרמטרים של הסימולטור אשר יכיל כתובת MAC, כתובת IP, מסיכה שתתאר את מרחב הכתובות הלוקליות ורשימה של הקשרים פתוחים (פורט מקור ופורט יעד). הארגומנט השני הוא קובץ עם כלל הפקטות, מופרדות ע"י ירידת שורה.

ממשו את המחלקה NIC_SIM, אשר מכילה 3 מתודות בלבד:

1. Constructor: מקבל כקלט שם של קובץ קורא ממנו את פרמטרי המערכת, ובהתאם לכך מעדכן את פרמטריו. קובץ הפרמטרים יהיה בפורמט TXT ויהיה בתבנית הבאה:

```
byte0:byte1:byte2:byte3:byte4:byte5
num0.num1.num2.num3/num4
src_prt:num0, dst_portb:num1
src_prt:num0, dst_portb:num1        ...
```

כאשר בשורה הראשונה תופיע כתובת ה-MAC של כרטיס הרשת בפורמט הקסדצימלי, בשורה השנייה כתובת ה-ip בפורמט עשורני ומסיכה שביחד מתארים את מרחב הכתובות הלוקלי.
מהשורה השלישית והלאה כל שורה תתאר תקשורת פתוחה של כרטיס הרשת ע"י זוג פורטים – מקור ויעד בפורמט עשרוני.

2. Flow: מקבלת כפרמטר שם של קובץ שמכיל פקטות מסוגים שונים, יעבור על כלל הפקטות יעדכן אותן וישמור אותן במקום הרלוונטי בזכרון. קובץ הפקטות יהיה קובץ TXT ויכיל בכל שורה פקטה שכתובה כמחרוזת.
3. Print_result: ידפיס את כלל הפקטות ב-RQ, לאחר מכן את כלל הפקטות ב-TQ ולבסוף את כלל המידע שנכתב להקשרים הפתוחים.

שימו לב - ניתן לראות ממשק מלא ותיעוד בקובץ nic_sim.hpp.

חלק ז' – בניית Makefile

יש לבנות קובץ Makefile אשר יקמפל את הקוד שמימשתם, לתכנית ההרצה יש לקרוא nic_sim.exe.

שימו לב: כשעובדים עם Cpp, הקונבנציה היא להשתמש במשתנה CXX במקום CC, וכן ב-CXXFLAGS במקום ב-CFLAGS.

עצרו! אם יש משהו שלא ברור בנוגע ל-Makefile – זה הזמן לשאול בפורום.

חלק ח' – הגשה:

דגשים מיוחדים:

1. ניתן להניח שהקבצים יהיו תקינים, וכן שפורמט הפקטות יהיה תקין.
2. אנו נשתמש בMakefile שלכם – באחריותכם לבדוק שהוא עובד כנדרש!
3. יש לוודא שהתוכנית פועלת ללא דליפות באמצעות valgrind. לתוכנית עם דליפות זכרון יורדו נקודות.
4. הקוד חייב לעמוד בקונבנציות הקוד כפי שראינו בתרגול.
5. שאלות בנוגע לתרגיל יש להפנות לפורום התרגיל בmoodle ובשעות הקבלה.

סיכום מפרט התרגיל:

קבצים נתונים	main.cpp, packets.hpp, NIC_sim.hpp, common.hpp
טסטים נתונים	test0_param.in test1_param.in test2_param.in test0_packets.in test1_packets.in test2_packets.in test0_res.out test1_res.out test2_res.out
קבצים להגשה	L4.h L4.cpp L3.h L3.cpp L2.h L2.cpp NIC_sim.h NIC_sim.cpp Makefile
שם הקובץ שצריך ליצור	nic_sim.exe

בהצלחה!!