האוניברסיטה העברית בירושלים

בית הספר להנדסה ולמדעי המחשב ע״ש רחל וסלים בנין

סדנת תכנות C ו- C++ - תרגיל 1

<u>נושאי התרגיל:</u> היכרות עם השפה, קומפילציה, משתנים, אריתמטיקה פשוטה, פלט, תנאים, לולאות, קבצים, פונקציות ושימוש ב- CLI

<u>תאריך הגשה:</u> 07.04.21 עד השעה 23:59

1 רקע

קריפטוגרפיה הוא תחום עתיק יומין שניתן למצוא תיעוד אליו עוד לפני מאות שנים. בעבר, נעשה שימוש בקריפטוגרפיה בעיקר על ידי הצבא והמלוכה, בעוד היום זהו נושא שניתן אף לטעון שכל אחד מאיתנו עושה בו שימוש על בסיס יום יומי, ואף בכל שניה שאנו משתמשים במחשב האישי שלנו (או במכשיר החכם הנייד) – אף מבלי לשים לב לכך.

בתרגיל זה נממש תוכנה המאפשרת להצפין ולקודד טקסט באמצעות צופן הנקרא ״צופן קיסר״ (נקרא גם ״צופן היסט״), ואף לבדוק את תקינותה.

2 צופן קיסר (צופן היסט)

נפתח בכך שנתאר כיצד פועל צופן קיסר באופן לא פורמלי: נסמן ב Σ את האלפבית ״שהמצפין״ יודע לקודד. המצפין מקבל מחרוזת כלשהי s וערך הזחה $k \in \mathbb{Z}$, עבור כל תו $c \in s$ אם $c \in \Sigma$ המצפין יבצע k הזחות של c בכיוון המתאים, לפי הסימן של k (חיובי יוזח ימינה ושלילי יוזח שמאלה).

למשל, אם k=2 וקיבלנו את התו 'A' אזי נזיח אותו פעמים – פעם ראשונה ל 'B' ופעם שניה ל 'C'. ערך 'C' הוא הערך שמתקבל, אפוא, מהצפנת התו 'A' עם k=2.

אם k הוא **שלילי** ההזחה תתבצע לכיוון השני. למשל, אם k=-2 וקיבלנו את התו 'D' אזי נזיח אותו פעמים – פעם ראשונה ל 'C' ופעם שניה ל 'B'.

עתה, ננסה להיות קצת יותר פורמלים, ונגדיר את צופן קיסר באופן הבא:

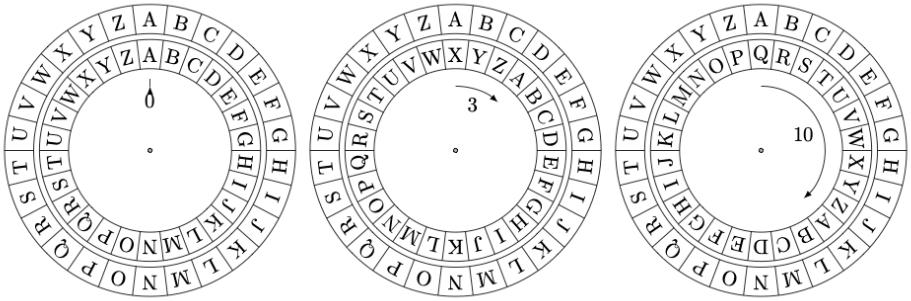
- יהי אלפבית Σ . תהי הפונקציה encode המקבלת 2 פרמטרים: מחרוזת לקידוד (הצפנה), שנסמנה - s ו- $k \in \mathbb{Z}$. encode מצפינה את s על ידי כך שעבור כל $c \in s$ המקיים $c \in \Sigma$ היא מבצעת k הזחות ציקליות.
  כשאומרים שהפעולה ציקלית, הכוונה היא לכך שהפעולה היא מעגלית – ולכן למשל,
  אם $\Sigma = \{'a', 'b', 'c'\}$ אזי הזחה ימינה של התו 'b' ב-2 מיקומים תחזיר את הערך 'a' לאור תכונת המעגליות. במילים אחרות, encode מזיזה k פעמים כל אות אלפביתית ב-s.
  לדוגמה באלפבית האנגלי:
  אם k=-2, אזי $'A' \mapsto 'Y', 'K' \mapsto 'I'$ בעוד אם k=2 אזי $'A' \mapsto 'C', 'a' \mapsto 'c'$.

- יהי אלפבית Σ . תהי הפונקציה decode המקבלת 2 פרמטרים: מחרוזת לפענוח, שנסמנה - s ו- $k \in \mathbb{Z}$. decode מפענחת את s על ידי כך שעבור כל $c \in s$ המקיים $c \in \Sigma$ היא מבצעת k הזחות ציקליות <u>בכיוון ההפוך לכיוון ההצפנה</u>.
  במילים אחרות, אם k חיובי decode מזיזה שמאלה k פעמים כל אות אלפביתית ב-s.
  לדוגמה: אם k=1, אזי $'b' \mapsto 'a', 'L' \mapsto 'K'$ ואם k=2 אזי $'c' \mapsto 'a', 'M' \mapsto 'K'$.
  אם k שלילי אז decode מזיזה ימינה k פעמים כל אות אלפביתית ב-s.
  לדוגמה: אם k = -1, אזי $'b' \mapsto 'c', 'L' \mapsto 'M'$.

כפועל יוצא מההגדרות הנ"ל, נניח כי תהי s מחרוזת ו- $k \in \mathbb{Z}$ ערך היסט, אזי נקבל

$$s = decode(encode(s))$$

לסיום, בתקווה שהדבר יפשט את הדברים, שימו לב לאילוסטרציה הבאה:



3 התוכנה cipher

בתרגיל זה נממש את התוכנה Cipher המאפשרת להצפין ולפענח קטעי טקסט באמצעות של צופן קיסר, ובנוסף לבדוק את תקינות של הצפנה.

3.1 קלט

התוכנה יכולה לבצע שלוש פעולות:

1) encode - להצפין מחרוזת מקובץ שמסופק לה.
2) decode - לפענח מחרוזת מקובץ המסופק לה.
3) check - לקבל שני קבצים, קובץ מקור וקובץ מוצפן, ולבדוק האם ההצפנה תקינה. במידה וכן, התוכנית תדפיס את פרמטר ההיסט (כלומר את ה k).

אם המשתמש בחר לבצע encode או decode התוכנית תקבל דרך ה(CLI (Command Line Interface **ארבעה** ארגומנטים בסדר הבא:

1. Command - ערך מסוג מחרוזת המציין את הפקודה המבוקשת. ערכי המחרוזת החוקיים יהיו "encode","decode" בלבד (עוד על בדיקות תקינות, בהמשך).
2. פרמטר ההיסט k – מציין את מספר ההזחות המבוקש (להצפנה/לפיענוח), כך ש- $k \in \mathbb{Z}$.
3. נתיב לקובץ קלט – בקובץ זה יהיה הטקסט שהמשתמש מבקש להצפין או לפענח.
4. נתיב לקובץ פלט – אל קובץ זה נכתוב את הטקסט לאחר הביצוע של ההצפנה או הפיענוח.

במידה והמשתמש בחר לבצע check התוכנית תקבל דרך ה (CLI (Command Line Interface **שלושה** ארגומנטים בסדר הבא:

1. Command - ערך מסוג מחרוזת המציין את הפקודה שרוצים לבצע, כלומר המחרוזת "check".
2. נתיב לקובץ המקור – בקובץ זה יהיה את הטקסט המקורי.
3. נתיב לקובץ המוצפן – בקובץ זה יהיה טקסט מוצפן (שהתוכנה תבדוק אם מוצפן באופן תקין, עוד על כך בהמשך).

3.1.1 קריאת הקלט ובדיקות תקינות

שימו לב לנקודות הבאות הנוגעות לקריאת הקלט:

- נזכיר שתוכלו לגשת לארגומנטים שהתקבלו מה CLI באמצעות argc, argv.
- לא ניתן לבצע השוואה בין מחרוזות באמצעות אופרטור ההשוואה (כלומר ==). כדי לבצע השוואה, תוכלו להשתמש בפונקציה המובנית strcmp. שימו לב שכדי להשתמש בפונקציה זו עליכם לכלול בראש התוכנית שלכם את הפקודה #include <string.h>.
- בתרגיל זה, **באופן חד פעמי**, נתיר את השימוש בפונקציה atoi על מנת להמיר מחרוזות למספר.[1]

כמו כן, שימו לב להנחות הבאות על הקלט:

- **אינכם רשאים** להניח כי מספר הארגומנטים שתקבלו בשורת הפקודה תקין (כלומר שלא קיבלתם פחות/יותר ארגומנטים מהנדרש).
- **אינכם רשאים** להניח כי הפקודה (Command) שקיבלתם אכן חוקית.
- **אינכם רשאים** להניח דבר על הטקסט שקיבלתם. בפרט, אינכם יכולים להניח כי הטקסט אינו כולל אותיות שאינן באלפבית האנגלי, שהטקסט אינו ריק וכדומה.
- **אינכם רשאים** להניח שהנתיב שקיבלתם לקובץ **הפלט** הוא של קובץ קיים, ועל כן:
  אם הוא קיים - יש לדרוס את הקובץ הקודם וליצור חדש. אם הוא לא קיים - יש לייצר קובץ חדש.
  כמובן שבשני המקרים יש לכתוב לתוך קובץ הפלט את הטקסט לאחר ההצפנה/הפענוח.
  (רמז: חשבו מהו מצב הפתיחה המתאים לכל קובץ).
- **ניתן להניח** כי הטקסט בכל אחד מהקבצים לא יעלה על 100,000 תווים.
- **ניתן להניח** להניח כי פרמטר ההיסט k הוא מספר שלם, אך **אינכם רשאים** להניח דבר על הערך והסימן שלו (מעבר לכך שהוא יכנס לטיפוס int).

---

[1] שימוש יותר נכון, חכם ובטוח יהיה למשל עם הפונקציה – strtol וכך נצפה שתעבדו בפעמים הבאות (אלא אם נאמר אחרת.)

3.1.2 טיפול בשגיאות

במקרים של שגיאה, עליכם להדפיס ל **stderr** הודעה אינפורמטיבית מהרשימה שלהלן ולצאת באופן מידי מהתוכנית עם קוד שגיאה (EXIT_FAILURE [1]):

שימו לב שעליכם לוודא שאתם סוגרים את הקבצים הפתוחים לפני היציאה מהתוכנית!

- אם כמות הארגומנטים שסופקה לתוכנית אינה תקינה, עליכם להדפיס את אחת מההודעות הבאות (בהתאם למקרה), המציינות כיצד צריך להיראות פורמט קלט תקין.

```
"Usage: cipher <encode|decode> <k> <source path file> <output path file>\n"
```

```
"Usage: cipher <check> <source path file> <output path file>\n"
```

- אם הפקודה שקיבלתם (Command) אינה תקינה, עליכם להדפיס את ההודעה:

```
"The given command is invalid\n"
```

- אם יש בעיה עם הקובץ (קובץ הקלט לא קיים/פתיחת הקובץ נכשלה), עליכם להדפיס את ההודעה:

```
"The given file is invalid\n"
```

במידה ויש מספר שגיאות, תודפס ל stderr הודעת שגיאה **אחת בלבד** הנבחרת לפי סדר החשיבות הבא:

1. פקודה (Command) לא תקינה.
2. מספר ארגומנטים לא תקין.
3. בעיה עם נתיב הקובץ או פתיחת הקובץ.

שימו לב: אם לא נשלחו כלל ארגומנטים, נתייחס לשגיאה כאל פקודה לא תקינה.

---

[1] EXIT_FAILURE – int קבוע שיושב בתוך קובץ header בשם stdlib ונוהגים להחזיר אותו במקרה של שגיאה. על מנת להשתמש בו יש להוסיף לתוכנית #include <stdlib.h>.

3.2 פלט

תוכנת ה cipher שלנו תצפין ותפענח רק אותיות שהינן באלפבית האנגלי, כל אות שאינה באלפבית תישמר כפי שהיא בפלט המוצפן.
במילים אחרות, במינוחים שראינו לעיל, נגדיר את האלפבית באופן הבא: $\Sigma = \{'A', 'B', \ldots, 'Z'\} \cup \{'a', 'b', \ldots, 'z'\}$.
כלומר, קבוצת כל האותיות בשפה האנגלית כך שיחס הסדר המוגדר עליהם הוא הסדר האלפבתי.
נשים לב שתכונות הציקליות שהזכרנו קודם, נשמרת עבור כל אחת מהקבוצות בנפרד (ראו דוגמה בהמשך).

עתה, בהנחה שלא היו שגיאות (כמפורט לעיל) התוכנה תפעל כך:

- אם הפקודה שהתקבלה היא encode: התוכנית תכתוב אל קובץ הפלט את ההצפנה של המחרוזת שבתוך קובץ הקלט, באמצעות האלגוריתם שהוצג לעיל. כמו כן, אין להדפיס ל stdout דבר או לכתוב אל תוך קובץ הפלט תוכן נוסף.
- אם הפקודה שהתקבלה היא decode: התוכנית תכתוב אל קובץ הפלט את הפיענוח של המחרוזת שהתקבלה, באמצעות האלגוריתם שהוצג לעיל. גם כאן, אין להדפיס ל stdout דבר או לכתוב אל תוך קובץ הפלט תוכן נוסף.
- אם הפקודה שהתקבלה היא check: התוכנית תבדוק אם ההצפנה תקינה, כלומר האם קיים $k \in \mathbb{Z}$ המקיים $k \in [0,25]$ שעבורו הטקסט בקובץ המוצפן הוא הצפנה תקינה של הטקסט בקובץ המקור. אם ההצפנה תקינה (כפי שהגדרנו לעיל) התוכנית תדפיס ל **stdout**:

```
"Valid encrypting with k = <value>\n"
```

לדוגמה: עבור הצפנה תקינה עם k = 2 יודפס: Valid encrypting with k = 2

שימו לב: עבור הצפנה תקינה של מחרוזת שכל תויה אינם שייכים לאלפבית (סמלים, מספרים וכו') יודפס: Valid encrypting with k = 0

במידה וההצפנה אינה תקינה התוכנית תדפיס ל **stdout**:

```
"Invalid encrypting\n"
```

בין אם ההצפנה תקינה ובין אם לא, במידה ואין שגיאה במהלך התוכנית ערך ההחזרה של התוכנית יהיה EXIT_SUCCESS.

3.3 דגשים והנחיות נוספות:

- נדגיש שוב כי כל אות בטקסט שאינה מופיעה ב $\sum$ תשאר כפי שהיא.
- נדגיש שוב כי ההזחות הציקליות מתקיימות בנפרד בין האותיות הגדולות ובין האותיות הקטנות. כלומר, כל אות גדולה תוצפן לאות גדולה וכל אות קטנה תוצפן לאות קטנה.
- זכרו כי פקודת המודולו (השארית) ב c המסומנת על ידי %, אינה תואמת לפקודת המודולו הנלמדת בשיעורי מתמטיקה.
- זכרו להשתמש בקבועים.
- נזכיר כי כדי להגדיר מערך סטטי נצטרך לדעת את גודלו בזמן קומפליציה.

על מנת להצהיר על קבוע שערכו ידוע בזמן קומפילציה, נוכל להשתמש בסינטקס הבא:

```
#define CONSTANT_NAME value
```

לדוגמא, נוכל להצהיר על הקבוע ARR_LENGTH עם הערך 10:

```
#define ARR_LENGTH   10
```

וכך נוכל להגדיר מערך סטטי בגודל 10:

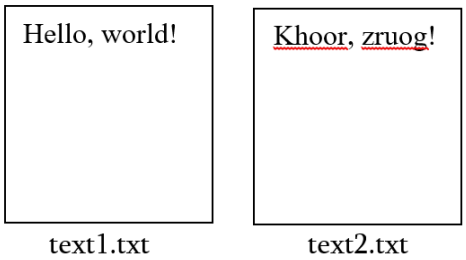
```
int arr[ARR_LENGTH];
```

- זכרו כי בכדי לרוץ על מערך בפונקציה נצטרך לדעת מהו האורך שלו (כלומר, להעביר אותו כארגומנט).
- זכרו לאתחל משתנים – קריאה/שימוש במשתנים שלא אותחלו עלול להוביל לתוצאות לא צפויות!
- אין להשתמש במשתנים גלובאלים ובהקצאות דינאמיות.
- הנכם רשאים ליצור פונקציות עזר כראות עינכם.
- הנכם רשאים לעשות שימוש בספריה הסטנדרטית של C (למרות שניתן בהחלט לפתור את התרגיל עם שימוש בפונקציות strcmp ו atoi בלבד).
- אינכם נדרשים להתייחס למצב שבו לקובץ הקלט ולקובץ הפלט יש את אותו הנתיב ואותו השם. כלומר, לא צריך להתייחס למקרה שבו קובץ הפלט דורס את קובץ הקלט.

4 דוגמאות

נפתח בדוגמה המדגימה את האופן שבו התוכנית מקודדת את הטקסט "Hello, world!" שנמצא בקובץ text1.txt עבור פרמטר ההיסט k=3 וכותבת את הפלט המוצפן אל תוך הקובץ text2.txt.

```
$ ./cipher encode 3 text1.txt text2.txt
```

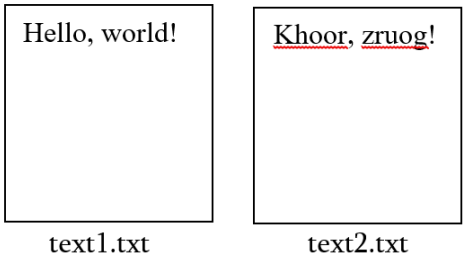


כעת, אם נרצה לפענח את הטקסט בקובץ text2.txt ("Khoor, zruog!") ולכתוב את הפיענוח אל תוך הקובץ text1.txt, נוכל להריץ את התוכנית עם הארגומנטים הבאים:

```
$ ./cipher decode 3 text2.txt text1.txt
```

כמו כן, עבור שורת הפקודה הבאה:

```
$ ./cipher check text1.txt text2.txt
```



יודפס ל stdout:

```
Valid encrypting with k = 3
```

<u>הערה:</u> סימן ה-$ המופיע בפקודות לעיל מסמן פקודה המבוצעת בשורת הפקודה (ב- Terminal).

5 נהלי הגשה

- קראו בקפידה את הוראות התרגיל ואת ההנחיות להגשת תרגילים שבמודל.
- זכרו כי התרגילים מוגשים ביחידים. אנו רואים העתקות בחומרה רבה!
- כתבו את כל ההודעות שבהוראות התרגיל בעצמכם. העתק-הדבק מקובץ התרגיל עלול להוסיף תווים מיותרים ולפגוע בבדיקה האוטומטית.
- בשפת C יש פונקציות רבות שעשויות להקל על עבודתכם. לפני תחילת העבודה על התרגיל, מומלץ לחפש באינטרנט את הפונקציות המתאימות ביותר לפתרון התרגיל. ודאו שכל הפונקציות שבהן אתם משתמשים מתאימות לסטנדרט C99, כי אתם יודעים כיצד הן מתנהגות בכל סיטואציה.
- יש להגיש את הפתרון בגיטהאב-האוניברסיטאי לפי נהלי ההגשה המוצגים כאן. תוכלו להעזר במדריכים שהועלו למודל ונגישים כאן וגם כאן.
- יש להגיש אך ורק את הקובץ cipher.c (אפשר להתעלם מקובץ ה README).
- כחלק מהבדיקה האוטומטית תיבדקו על סגנון כתיבת קוד.
- כדי להדר את התרגיל מהקובץ cipher.c לקובץ בינארי בשם cipher תוכלו להשתמש בפקודה הבאה:

```
gcc -Wextra -Wall -Wvla -std=c99 -lm cipher.c -o cipher
```

- **שימו לב -** הבדיקות האוטומטיות רצות על גבי מחשבי בית הספר, לכן וודאו כי הפתרון שלכם רץ ועובד על גבי מחשבי בית הספר (תוכלו לבדוק זאת גם באמצעות חיבור מרחוק).
- **שימו לב -** ודאו כי הפתרון שלכם עובר את הפריסאבמיט ללא שגיאות או אזהרות, כשלון בקומפילציה או בפריסאבמיט יגרור ציון 0 בתרגיל. למעוניינים, ניתן להריץ את בדיקת הפריסאבמיט על קובץ tar המכיל את ההגשה שלכם באמצעות הפקודה הבאה (במחשבי האקווריום או בחיבור מרחוק בלבד):

```
~labcc2/presubmit/run <path_to_tar_submission>
```

בהצלחה!