

האוניברסיטה העברית בירושלים

בית הספר להנדסה ולמדעי המחשב ע"ש רחל וסלים בנין

סדנאות תכנות בשפת C++ ו-C++ - קיץ (קורס 67320) C++ - תרגיל 2

תאריך ההגשה של התרגיל והבוחן התיאורטי: יום רביעי, ה-18 בספטמבר, 2019 - עד השעה 23:55;
הגשה מאוחרת (בהפחתת 10 נקודות): יום חמישי, ה-19 בספטמבר, 2019 - עד השעה 23:55.

נושאי התרגיל: smart point-, exceptions, C++ standard library, templates, ers.

1 רקע

מימוש של תיבת דואר אלקטרוני הוא למעשה מימוש של פרוטוקול, המאפשר לשני מחשבים (דהיינו, שרתים) "המדברים" באותו הפרוטוקול לשלוח ולקבל הודעות.¹ עובדה זו מאפשרת לשלוח בקלות הודעות המכילות מסרים "זדוניים", שמטרתם לפגוע במשתמש הקצה (בין אם על ידי שליחת וירוסים, קישורים שמאפשרים גניבת זהות, בקשות להעברת כסף וכיוצ"ב). לכן, כדי למנוע חשיפת משתמשים לסיכונים אלו - נרצה ליצור מנגנון המאפשר "לקטלג" (to classify) הודעות שנדמה כי הן "זדוניות", ולהסב את תשומת לבו של המשתמש לכך. מנגנון זה מוכר בשם Spam Detector.

בפן הטכני, אחת השיטות הפשוטות למימוש מנגנון מסוג זה היא יצירת בנק, או מאגר, של "ערכים רעים" שנרצה לסנן. לכל ערך כאמור נצמיד ניקוד המצביע על חומרת השימוש בערך. אילו הודעה עברה סף מסוים, שמסמן את סכימת הניקוד של כל "הערכים הרעים" שהופיעו בהודעה - אזי מדובר בהודעת Spam.

בתרגיל זה נשתמש בידע שצברנו במהלך הקורס ונשלבנו בנושאים כדוגמת גרירות ושימוש בספרייה הסטנדרטית של C++, לצורך מימוש מבנה הנתונים HashMap ושימוש בו. בחלק הראשון של התרגיל, נממש את מבנה הנתונים HashMap, בעוד בחלק השני - נממש Spam Detector לפי העיקרון האמור.

2 הגדרות

להלן נזכיר מספר הגדרות בסיסיות הנוגעות לטבלאות גיבוב, שהוצגו בקורס מבני נתונים:

¹פרוטוקול מוכר, למשל, הוא POP - Post Office Protocol - https://en.wikipedia.org/wiki/Post_Office_Protocol

- **הגדרה:** קבוצה (set) זו קבוצה של איברים שאינה מסודרת בסדר כלשהוא. סט אינו חייב להכיל ערכים מתמטיים, אלא עשוי להכיל ערכים מסוגים שונים. במתמטיקה נהוג לסמן קבוצה זו עם סוגרים מסולסלות (למשל $\{1, 5, 8\}$ או $\{Yossi, Dana\}$, וכן $\{1, 5, 8\} = \{1, 8, 5\} = \{5, 8, 1\}$).
כשנדבר על מימוש של קבוצה בקוד, נרצה לממש לכל הפחות את הפעולות `add(...)`, `contains(...)`, `delete(...)` ו-`size(...)`. נזכיר שאלו פעולות סטנדרטיות במבני נתונים בספריה הסטנדרטית של C++, כדוגמת `std::vector`.
- **הגדרה:** פונקציה גיבוב (hash function) היא פונקציה שממפה מידע מגודל כלשהוא ("מפתחות") למידע אחר כלשהוא, בגודל סופי. לשם הפשטות, נוכל להניח שמדובר בפונקציה מהצורה $h : U \rightarrow \{0, \dots, m-1\}$ כאשר U היא קבוצת איברים כלשהיא, כמו למשל מחרוזות, מספרים וכדומה, ו- $\{0, \dots, m-1\}$ היא קבוצה סופית של תוצאות אותן ניתן לקבל מהפונקציה (תוצאת פעולת ה-hash).
- **מסקנה מההגדרה הקודמת:** נרצה שכל פונקציה hash, שנסמנה h , תקיים:
 - h תהיה קלה לחישוב;
 - h -ש תמפה כמה שפחות איברים לאותו התא, כדי למנוע התנגשויות רבות.
- **הגדרה:** hash map הוא מבנה נתונים המכיל מיפוי של מפתחות לערכים. המפתחות יכולים להיות מספרים, מחרוזות או כל טיפוס נתונים נתמך אחר – וכך גם הערכים. ברמה האינטרנית, טיפוס נתונים זה עושה שימוש בטבלת גיבוב כדי **למפות** מפתחות לערכים. הייתרון של מבנה נתונים זה, הוא שבהינתן פונקציה גיבוב "טובה", פעולות ההוספה, החיפוש וההסרה שלו מבוצעות בסיבוכיות זמן ריצה ממוצעת של $\Theta(1)$.

3 מימוש HashMap

משהוצגו ההגדרות המקדימות הנדרשות לפתרון התרגיל, נציג להלן מספר נושאים נוספים הנוגעים לפונקציות, טבלאות ומפות גיבוב, להם אתם נדרשים במימוש התרגיל:

3.1 גורם העומס (Load Factor)

לבד מגודל הקלט בפועל, הביצועים של מפת הגיבוב מושפעים משני פרמטרים: גורם עומס העליון וגורם העומס התחתון (upper load factor & lower load factor). גורם העומס מוגדר כך:

$$Load\ Factor = \frac{M}{capacity}$$

כאשר $M \in \mathbb{N} \cup \{0\}$ מייצג את כמות האיברים שמבנה הנתונים מכיל כרגע – בפועל, בעוד $capacity \in \mathbb{N} \cup \{0\}$ מייצג את כמות הנתונים **המקסימלית** שניתן לשמור במבנה הנתונים (כלומר, הקיבולת). אם כך, מהם גורמי העומס התחתון והעליון? אלו הגורמים שמציינים עד כמה נסכים שמבנה הנתונים יהיה ריק או ממלא. כלומר, נרצה שכאשר נחצה רף מסוים (threshold) – נגדיל או נקטין את הטבלה בהתאם, כך שמצד אחד לא תדרוש זיכרון רב מדי ומהצד השני תוכל להכיל כמה איברים שנרצה.
לשם כך, עת שנחצה את אותו רף (תחתון או עליון) – נבצע הליך שנקרא re-hashing ובו נגדיל או נקטין את הטבלה, נחשב את ערכי ה-hash בשנית ונמקם שוב את כל האיברים שבטבלה. במילים אחרות, ננעתיק את כל הערכים הישנים לטבלה "חדשה".
כברירת מחדל, נגדיר את גורם העומס התחתון להיות $\frac{1}{4}$ ואת גורם העומס העליון להיות $\frac{3}{4}$.

3.2 גודל הטבלה

ישנן שתי אפשרויות פופולריות לבחירת הגודל המקסימי של טבלאות גיבוב: שימוש במספרים ראשוניים או בחזקות של 2. האופציה הראשונה, דהיינו מספרים ראשוניים, טובה מכיוון שפיזור האיברים נעשה בצורה הרבה יותר אחידה. מנגד, הקושי שבבחירה זו הוא שאין דרך קלה לבצע re-hash – כלומר לא נוכל, למשל, לבצע חישוב אריתמטי פשוט כמו העלאה בחזקה, כדי לקבל את גודל הטבלה החדש. מנגד, בעוד שהאופציה השנייה – דהיינו שימוש בחזקות של 2 – אינה יוצרת פיזור אחיד טוב מספיק, מאוד קל לבצע בעניינה re-hashing וכן ניתן לחשב בה את המיקום של כל איבר בדרך מהירה יותר, על ידי bitwise operators. לפיכך, בתרגיל זה, גודל הטבלה יהיה חזקה של 2, כאשר הגודל ההתחלתי הוא 16.

3.3 פונקצית הגיבוב

כאמור לעיל, עלינו לבחור פונקצית גיבוב "טובה" כדי להגיע למבנה נתונים שפועל ביעילות טובה. פונקצית הגיבוב שנבחר מאוד בסיסית, והיא:

$$h(x) = x \bmod size, \quad size \in \mathbb{N} \cup \{0\}$$

כאשר $size$ מייצג את גודל הטבלה ו- x הוא ייצוג מספרי של הערך שנרצה לשמור בטבלה. כדי להמיר מחרוזת, שבר וכיוצ"ב למספר שלם, נוכל להשתמש בפונקציה `std::hash`. עתה, נשים לב שאופרטור ה-`modulo` ב-`C++` אינה עובדת באופן זהה לזה שהכרנו במתמטיקה. כך למשל, $-3 \bmod 7 = -3$, בעוד שהתוצאה המתמטית אמורה להיות 4. לכן, כדי לפתור זאת, יהיה עלינו לחשב את הפונקציה ב-`C++` כערך מוחלט. יתרה מכך, אנו עשויים להיתקל בקושי כאשר נחשב את המודולו של `INT_MIN`, כיוון שאין לו ערך מקסימלי תואם. נוכל לפתור זאת על ידי `long`-casting. **לסיים, נציע חלופה אחרת לחישוב פונקצית הגיבוב:** נבחין שפונקצית הגיבוב שהגדרנו עושה שימוש ב-`size`, שכאמור לעיל הוא חזקה של 2. לכן, נוכל להשתמש באופרטור הלוגי `and` (מיוצג על ידי `&`) כדי לחשב את אותו הערך, כלומר:

$$v \bmod size = v \& (size - 1)$$

פתרון זה עדיף, כיוון שאופרטורים לוגים מהירים יותר מאריתמטים – אז חישוב האינדקס יהיה מהיר יותר. בנוסף, גם לא ניזקק יותר לשימוש בערך מוחלט או ל-`long`-casting.

3.4 מיפוי הגיבוב – שיטת מיפוי פתוח (Open Hashing).

בהמשך לאמור, קל לראות שפונקצית הגיבוב שנבחרה, $h(x)$, תיצור, במוקדם או במאוחר, התנגשויות. ראשית, אם כמות האיברים שנרצה לשמור במבנה הנתונים גדול מ- $size$, זה נובע ישירות מעיקרון שובך היונים. אחרת, גם כאשר $size$ גדול מכמות האיברים שנרצה לשמור, אנו עשויים להיתקל במקרים בהם לשני ערכים, x, y *s.t.* $x \neq y$, נקבל ש- $h(x) = h(y)$ ובמקרה זה ניתקל בהתנגשות. ישנן שתי שיטות לפתרון התנגשויות:

- **Open hashing:** שיטה המאפשרת לשמור יותר מערך אחד בכל תא. התאים, שנקראים "סלים" (buckets) ובנויים מטיפוס נתונים אחר – לדוגמה מרשימה מקושרת. כך, גם אם יש התנגשות, כל שקורה הוא שהאיבר המתנגש נוסף לרשימה המקושרת.
- **Close hashing:** שיטה לפיה כל תא יכול להכיל רק איבר אחד. במקרים אלו, עלינו למצוא דרך אחרת להתמודד עם התנגשויות ולמפות איברים.²

²שיטה מוכרת למימוש `closed hashing` היא `quadratic probing`. להרחבה ראו: https://en.wikipedia.org/wiki/Quadratic_probing.

בתרגיל זה, נממש את ה־hash set באמצעות Open hashing.

4 חלק א' – המחלקה HashMap

בחלק הראשון נממש את המחלקה הגנרית HashMap, שתיוצג מיפוי בין מפתחות לערכים $(key \mapsto value)$, המבוסס על טבלת גיבוב. כלומר, מבנה הנתונים ימפה בין מפתחות, מסוג KeyT, לערכים, מסוג ValueT. יש לתמוך ב-API הבא:

הערות	התיאור	
פעולות מחזור החיים של האובייקט		
	בנאי שמאתחל HashMap ריק.	בנאי ברירת מחדל
הערכים מטיפוס double.	בנאי המקבל פקטור עומס תחתון ועליון.	בנאי 1
יש לוודא שהוקטורים באותו הגודל. המיפוי יהיה $\forall 0 \leq i < n \text{ keys}[i] \mapsto \text{values}[i]$.	בנאי המקבל שני וקטורים, אחד שמכיל ערכי KeyT ואחד שמכיל ערכי ValueT ושומר את הערכים במפה לפי הסדר.	בנאי 2
	מימוש של בנאי העתקה.	בנאי העתקה
	מימוש של בנאי הזחה.	move constructor
	מימוש destructor.	destructor
פעולות		
המספר שמחזור מטיפוס int.	הפעולה מחזירה את כמות איברי המפה.	size
המספר שמחזור מטיפוס int.	פעולה המחזירה את קיבולת המפה.	capacity
המספר שמחזור מטיפוס double.	פעולה המחזירה את גורם העומס.	getLoadFactor
הפעולה תחזיר bool.	פעולה הבודקת האם המפה ריקה.	empty
הפעולה תחזיר אמת אם הערך נוסף בהצלחה.	פעולה המקבלת מפתח וערך, ושומרת את המיפוי שהתקבל.	insert
הפעולה מחזירה bool.	הפעולה מקבלת מפתח ובודקת האם הוא קיים במפה.	containsKey
הפעולה תזרוק חריגה במקרה שה־key לא נמצא.	פעולה מקבלת key ומחזירה את ה־value המשווייך אליו.	at
הפעולה תחזיר אמת אם הערך הוסר בהצלחה.	הפעולה מקבלת מפתח ומסירה את הערך המשווייך לו מהמפה.	erase
הפעולה תחזיר int.	תקבל מפתח ותחזיר את גודל הסל.	bucketSize
	פעולה המסירה את כל איברי המפה.	clear
עליכם לממש const forward iterator בלבד. ה־iterator יחזיר std::pair<KeyT, ValueT>.	מימוש כל הפעולות הנדרשות ל־iterator (לרבות typedefs), בהתאם לשמות הסטנדרטים של C++.	iterator
אופרטורים		
השמה לכל ערכי האובייקט.	תמיכה באופרטור ההשמה (=).	השמה
האופרטור יקבל מפתח ויחזיר את הערך המשווייך לו. אין לזרוק חריגה במקרה זה (no-throw guarantee).	תמיכה באופרטור [].	subscript
בדיקה האם שני סטים מכילים איברים זהים.	תמיכה באופרטורים ==, !=.	השוואה

דגשים, הבהרות, הנחיות והנחות:

- את המחלקה עליכם להגדיר בקובץ `HashMap.hpp` (למה יהיה קושי להגדירו עם קובצי `h` ו-`cpp`?).
- כאמור, על המחלקה להיות גנרית. הערך הגנרי הראשון שהמחלקה תקבל הוא טיפוס הנתונים שמייצג את המפתחות, אליו התייחסנו בשם `KeyT`. הפרמטר השני הגנרי שהמחלקה תקבל הוא סוג הנתונים המייצג את הערכים אליהם המפתחות ממפים, לו קראנו `ValueT`. **ניתן להניח** ש-`KeyT` נתמך על ידי `std::hash`.
- **שימו לב:** ה-API הנ"ל מציג לכם את שמות הפונקציות המחייבות, הפרמטרים, ערכי החזרה וטיפוסייהם. בעת מימוש ה-API, עליכם ליישם את העקרונות שנלמדו בקורס באשר לערכים קבועים (`constants`) ומשתני ייחוס (`references`). **שימוש בקונבנציות אלו הוא חלק אינטגרלי מהתרגיל, עליו אתם מקבלים ניקוד. עיקרון זה נכון בפרט גם למימוש ה-`iterator`.**
- בתרגיל זה אתם נדרשים לעשות שימוש ב-STL. בחירת מבני הנתונים המתאימים לאחסון המידע, מתוך STL, וכך גם לביצוע חישובים נדרשים מהווה חלק אינטגרלי מהתרגיל. עם זאת, ומבלי לסתור את ההוראה שלעיל בוגע STL, אתם כן **רשאים** ליצור מחלקה או מחלקות עזר, ככול שהן נמצאות בקובץ הנ"ל ועונות לדרישות התרגיל. שימו לב לנמק את בחירותיכם ב-`README`.
- **מסקנה מהאמור – אין להשתמש במימושים הדיפולטיבים של STL בעת מימוש המחלקות. בפרט, וודאי שאסור, למשל, להשתמש ב-`std::map`.**
- על הפעולות `insert`, `at`, `contains`, `erase` (וכמובן `operator[]`) לפעול בסיבוכיות לינארית ביחס לסל שבו האיבר נמצא – כלומר ב- $O(n)$ כאשר n הוא גודל הסל.
- לא ניתן להניח שהמפתחות שווים לערכים (כלומר גם אם `KeyT = ValueT`, לא בהכרח עבור i כלשהו $keys[i] = values[i]$).
- לא ניתן להניח שהערכים שהתקבלו תקינים. בפרט, הערכים שמתקבלים עשויים להכיל hash code זהה.
- **חידוד:** יש להבדיל שבין `HashMap::at` ובין `HashMap::operator[]`. בעוד בראשון תיזרק חריגה במקרה שהאיבר לא קיים, בשני זה לא המצב. **במקרה שבו פונים ב-`HashMap::operator[]` לאיבר שלא קיים, עליכם ליצור איבר חדש בטבלה.** גישה זו תאפשר לנו להשתמש בביטויים כמו `"bar" = map["foo"]`. מנגד, אם המתכנת ינסה לפנות לאיבר שלא קיים למטרות קריאה – ההתנהגות תהיה לא מוגדרת ותלויה בכס.³ במילים אחרות, מצופה מהמתכנת שלא לגשת לערכים שלא קיימים, למטרות קריאה, דרך `HashMap::operator[]`.

³התנהגות זו תואמת להתנהגות של `std::map`. ראו למשל: [http://www.cplusplus.com/reference/map/map/operator\[\]/](http://www.cplusplus.com/reference/map/map/operator[]/). למעשה, כך עובדים גם טיפוסים נתונים אחרים ב-STL. למשל ב-`std::vector` מוחזר מקום שלא מוגדר בזיכרון.

5 מימוש SpamDetector

ניגש עתה לחלק השני של התרגיל, והוא – כאמור – מימוש מנגון לזיהוי הודעות Spam. את התוכנית נחבר בקובץ SpamDetector.cpp ופעולתה תהא זו: התוכנית תקבל שני קבצים דרך ה-CLI: קובץ "database" של מילים רעות וקובץ המכיל את התוכן של ההודעה שרוצים לנתח. כמו כן, התוכנית תקבל ב-CLI גם ארגומנט שלישי, והוא רף הניקוד (threshold) שחצייתו מסמנת כי מדובר בהודעת Spam. לאחר בניית אובייקט שמייצג את ה-database, התוכנית תעזר ב-database כדי לקטלג האם ההודעה חשודה כספאם אם לאו. להלן נגדיר בפירוט כיצד יתבצע כל שלב:

5.1 קלט

5.1.1 קובץ ה-database

קובץ ה-database יהיה בפורמט CSV (Comma-separated values)⁴. בקובץ, כל שורה תתאר צירוף רע ואת הניקוד שלו:

0 תא	1 תא
הניקוד הניתן לצירוף	הצירוף הרע

הערות והנחיות:

- על אף שפורמט הקובץ הוא CSV, הסיומת של הקובץ לא חייבת להיות .csv. ויכולה להיות כל סיומת אחרת, למשל .txt או .in.
- **לא ניתן** להניח שהקובץ קיים או שהקובץ אינו ריק. אם הקובץ לא קיים – מדובר בשגיאה. אם הקובץ ריק – ניתן להתקדם הלאה (פשוט לא קיימים ביטויים "רעים").
- **ניתן** להניח שמדובר בקובץ CSV תקין. **מנגד, לא ניתן** להניח שיהיו רק שתי עמודות – עליכם לוודא זאת. קובץ שאינו עומד בפורמט המתואר **נחשב קובץ פגום** ועליכם לפעול בהתאם להוראות שיוצגו בהמשך (בפרק של טיפול בשגיאות).
- **לא ניתן** להניח שערכי העמודות תקינים. אם אחד מהערכים אינו תקין – מדובר בקובץ פגום. בפרט:
 - אם לא נשלח אף ערך לתא ה-0 (למשל הוץ "5") – מדובר בקובץ פגום.
 - ביטוי יכול להכיל יותר ממילה אחת. **ניתן** להניח שבצירוף לא יופיע פסיק (כדי לא לפגוע בתקינות הפורמט). עם זאת, **לא ניתן להניח** שלא יופיעו תווים מיוחדים אחרים, למשל > או <.
 - עליכם לוודא שערך הניקוד יהיה חיובי או 0. אם הערך שלילי, מדובר בשגיאה.
 - **שימו לב:** **ניתן להניח** שלא יהיו ביטויים שמוכללים אחד בשני. לדוגמה: אם מופיע הביטוי "your bank account details", אזי לא יופיעו בקובץ ביטויים כדוגמת "bank account" או "account details" – כדי למנוע חפיפה. מנגד, כן יכול להופיע הביטוי "forward me your bank account" כיוון שהוא אינו מוכל במלואו בביטוי הראשון (אין ביניהם חפיפה חלקית).

- הנכם **רשאים** להשתמש בספריית filesystem⁵ של boost⁶ (ובה בלבד).

⁴ניתן להכיר את הפורמט ולראות דוגמה כאן: https://en.wikipedia.org/wiki/Comma-separated_values

⁵ראו: https://www.boost.org/doc/libs/1_70_0/libs/filesystem/doc/index.htm

⁶תוכלו לקרוא על הספרייה המופלאה הזו כאן: [https://en.wikipedia.org/wiki/Boost_\(C%2B%2B_libraries\)](https://en.wikipedia.org/wiki/Boost_(C%2B%2B_libraries))

- הנכם **רשאים** לעשות שימוש ב-STL (C++ Standard Template Library).

5.1.2 קובץ הקלט (תוכן הודעת הדואר האלקטרוני)

הערך השני שתקבלו דרך ה-`cli` יהא נתיב לקובץ הקלט, שמייצג את התוכן של הודעת הדואר האלקטרוני. תוכן זה יופיע כטקסט פשוט (Plain Text) ואותו עליכם לנתח. שימו לב:

- **לא ניתן** להניח שהקובץ קיים או שהקובץ אינו ריק. אם הקובץ לא קיים – מדובר בשגיאה; מנגד, אם הקובץ ריק – וההודעה תקינה “באופן ריק”.
- גם כאן הנכם **רשאים** להשתמש בספריית `filesystem` של `boost` (ובה **בלבד**) וב-STL.

5.1.3 רף הניקוד המקסימלי לזיהוי Spam – `threshold`

לבסוף, הנכם נדרשים לקלוט מה-`cli` ערך מספרי **חיובי ממש** שמסמן את הרף לזיהוי הודעת Spam. לא ניתן להניח שקלט זה יהיה תקין באף מובן.

5.2 מימוש ופלט

עתה, על תוכניתכם לנתח את ההודעה באמצעות קריאתה ולבחון האם היא מסווגת כ-Spam או לא. הפלט שעל תוכניתכם להדפיס יהיה “SPAM” אם מדובר בספאם, ו-“NOT_SPAM” אם לא מדובר ב-Spam בלבד. בסוף ההדפסה, כמובן, תבוא ירידת שורה. שימו לב:

- עליכם לבדוק האם הניקוד הכולל של הקובץ חוצה או שווה ל-`threshold` (כלומר $threshold \leq score$). אם התשובה לכך חיובית – זהו ספאם.
- עליכם לעשות שימוש ב-`HashMap` שיצרתם לשם מימוש חלק זה של התרגיל. כדאי, אך לא חובה, להיעזר ב-`iterator` שכתבתם בעת הבדיקה.
- **חשוב:** עליכם לבצע את כל ההשוואות בין המילים שבקובץ ה-`database` ובין הטקסט **באותיות קטנות**. לכן, אם המילה “Facebook” מופיעה (כך ממש) ב-`database`, היא תילכד בכל המקרים “facebook”, “FaceBook”, “FACEBOOK”, “Facebook”.

5.3 רהתמודדות עם שגיאות

עליכם לטפל במקרים בהם לא התקבל קלט תקין. אם מספר הארגומנטים שנשלחו לתוכנה אינו תקין, עליכם להדפיס ל-`stderr` את הפלט:

```
Usage: SpamDetector <database path> <message path> <threshold>\n
```

מנגד, אם נתקלתם בקלט שגוי (קובץ שאינו קיים, $threshold \leq 0$ וכו') – עליכם להדפיס:

```
Invalid input\n
```

בשני המקרים “\n” מסמן ירידת שורה (כפי שנהוג להדפיסה ב-C++). לאחר הדפסת הפלט, בשני המקרים עליכם לסגור באופן מיידי את התוכנית עם קוד סיום `EXIT_FAILURE`. **שימו לב:** עליכם להתמודד עם כל חריגה שתוכניתכם עלולה לייצר, בין אם בעקבות חריגה שאתם זרקתם באופן מפורש, ובין אם בעקבות חריגה ש-C++ עלולה לזרוק. ההודעה שיש להדפיס תהיה תלויה בסיבה לחריגה.

6 דוגמה

נניח ש־database.csv מכיל את התוכן הבא:

```
financial help,5
billionaires,10
millionaires,10
lucky,10
random,5
your bank account details,20
Facebook username and password,50
```

וכן נניח ש־message.txt מכיל את התוכן הבא:

Winner found!

Hello there, I'm Mark Zuckerberg, the founder and CEO of the social-networking website Facebook, as well as one of the world's youngest billionaires. I decided to secretly give \$1,500,000.00 selected individuals worldwide at random. You should count yourself as the lucky individual. Your email address was chosen online while searching at random. Kindly email me back with your bank account details and Facebook username and password so that I can verify your identity and send you the money.

Kind regards,
Mark Zuckerberg

אם כן, נשים לב שסריקת הקובץ טוביל לתוצאה הבאה:

צירוף רע	כמות הופעות	ניקוד כולל (ניקוד כפול הופעות)
financial help	0	0
billionaires	1	10
millionaires	0	0
random	2	10
lucky	1	10
your bank account details	1	20
Facebook username and password	1	50

בסך הכול, קיבלנו שניקוד ההודעה הוא 100. לכן נקבל בהרצת SpamDetector:

```
$ SpamDetector database.csv message.txt 50
SPAM
$ SpamDetector database.csv message.txt 150
NOT_SPAM
```

אשר שורה שנפתחת ב־\$ מסמנת את הפקודה שהוקלדה.

7 נהלי הגשה

- קראו בקפידה את הוראות תרגיל זה ואת ההנחיות להגשת תרגילים שבאתר הקורס.
- זכרו שהחל מתרגיל זה עליכם לקמפל את התוכנית כנגד מהדר לשפת C++ בתקן שנקבע בקורס. כמו כן, זכרו שעליכם **לתעדף** פונקציות ותכונות של C++ על פני אלו

של C. למשל, נעדיף להשתמש ב-new ו-delete על פני malloc ו-free, וכן נעדיף להשתמש ב-std::string מאשר ב-char*.

- **נזכיר:** כאמור בהנחיות הכלליות להגשת תרגילים - הקצאת זיכרון דינמית מחייבת את שחרור הזיכרון, למעט במקרים בהם ישנה שגיאה המחייבת סגירת התוכנית באופן מיידי עם קוד שגיאה (כלומר קוד יציאה השונה מ-0). תוכלו להיעזר בתוכנה valgrind כדי לחפש דליפות זיכרון בתוכנית שכתבתם.

- פתרון בית הספר זמין בנתיב

~proglab/www/cpp_ex3/SpamDetector

- עליכם ליצור קובץ tar הכולל את הקבצים SpamDetector.cpp, HashMap.hpp. README ניתן ליצור קובץ tar כדרוש על ידי הפקודה:

```
$ tar -cvf cpp_ex3.tar HashMap.hpp SpamDetector.cpp README
```

שימו לב: קבצי קוד המקור שתכתבו נדרשים להתקמפל כהלכה עם ++14, std, כנדרש בהוראות להגשת תרגילים שפורסמו באתר הקורס.

- אנא וודאו כי התרגיל שלכם עובר את ה-Pre-submission Script **ללא שגיאות או אזהרות**. קובץ ה-Pre-submission Script זמין בנתיב.

~proglab/www/cpp_ex3/presubmission

בהצלחה!!