אלגוריתמים, סמסטר ב' תשפ"ה,
תרגיל תכנותי

מועד הגשה: עד ליום א' 1.6

הנחיות כלליות:

א. התרגיל הינו תרגיל חובה.

ב. התרגיל ניתן להגשה בבודדים או בזוגות, אך לא בקבוצות גדולות יותר.

ג. התוכנית תיכתב בשפת C++ (ובה בלבד. בחירת השפה אינה נתונה לשיקול דעת הסטודנטים).

ד. גם תלמידים החוזרים על הקורס מחויבים בהגשת התרגיל.

ה. איחור בהגשה יאושר רק במקרים חריגים וגם זאת רק בתנאי שהפנייה למרצה בנושא נעשתה לפני מועד ההגשה המקורי של התרגיל.

נושא התרגיל: מימוש **אלגוריתם מילוי הכדים** שנלמד בכיתה.

שלבים לביצוע התרגיל:

א. קראו תחילה היטב את ההנחיות של הפרויקט והבינו מה נדרש בדיוק.

ב. תכננו את ה-design של התוכנית שלכם: בחרו אילו מחלקות תממשו, החליטו על data members מתאימים ועל methods רלוונטיים לכל מחלקה.

ג. כתבו מימוש מלא לכל המחלקות שייעשה בהן שימוש במסגרת התוכנית.

ד. ממשו את האלגוריתמים הנדרשים ואת פונקציית ה-main של התוכנית.

תיאור הבעיה:

במטלה זו נראה שני מימושים שונים למציאת לבעיית מילוי הכדים שנלמדה בכיתה.

תזכורת:

נתונים 2 כדים, כד אחד קטן שנפחו S ליטרים וכד אחד גדול שנפחו L ליטרים (L > S).

בנוסף נתון נפח W בליטרים.

במצב ההתחלתי 2 הכדים ריקים.

ניתן לבצע מספר פעולות שיפורטו בהמשך, כאשר המטרה היא להגיע למצב שבו הכד הגדול מכיל W ליטרים של מים בדיוק, והכד הקטן ריק.

הפעולות שאותן ניתן לבצע הן:

1. מילוי כל אחד מהכדים במים מהברז, עד שהכד מלא לגמרי.
2. ריקון כל אחד מהכדים לכיור, כך שהכד המרוקן יכיל 0 ליטרים.
3. העברת מים מכד אחד לכד אחר, עד אשר הכד ממנו הועברו המים התרוקן לחלוטין או שהכד אליו הועברו המים התמלא לחלוטין (הראשון מבין השניים).

שימו לב, שלא תמיד ניתן להגיע למצב הסופי!

קלט האלגוריתם: שלושה מספרים טבעיים L, S, W כך ש- $L > S$ וגם $W \leq L$.

פלט האלגוריתם: אם ניתן להגיע למצב הסופי הרצוי, הפלט יורכב מ:

- מספר הפעולות המינימאלי שיש לבצע על מנת להגיע מהמצב ההתחלתי (שבו שני הכדים ריקים) למצב הסופי (שבו בכך הגדול יש W ליטרים והכד הקטן ריק).
- סדרת הפעולות המינימאלית כפי שנתן האלגוריתם
(שימו לב שלצורך טסטים אחידים נקפיד על מיוון לקסיקוגרפי כמפורט בהמשך)

אחרת – הודעה שלא ניתן להגיע למצב הסופי הרצוי.

<u>תיאור התכנית הראשית :</u>

1. התוכנית תקלוט מהמשתמש 3 מספרים טבעיים : L, S, W (משמאל לימין).
2. התוכנית תקלוט מספר טבעי נוסף מהמשתמש, המייצג את המימוש שיבוצע בריצה זו (1 או 2).
3. התוכנית תקלוט בנוסף מספר טבעי מהמשתמש האם ברצונו לראות את זמן הריצה של הפונקציה. 0 – אינו מעוניין, 1 - מעוניין
4. התוכנית תיישם את המימוש שנבחר ותדפיס למסך את הפלט הנדרש.

<u>כך למשל קלט תקין ייראה כך :</u> 4 3 2 1 0

כלומר : L = 4, S = 3, W = 2, המימוש הנבחר הוא המימוש הראשון והמשתמש אינו מעוניין בזמן הריצה של הפונקציה

<u>במקרה זה הפלט יהיה :</u>

Number of operations: 6
Operations:
1. Fill small jug
2. Transfer from small jug to large jug
3. Fill small jug
4. Transfer from small jug to large jug
5. Empty large jug
6. Transfer from small jug to large jug

<u>דוגמה נוספת לקלט תקין :</u> 6 4 5 2 1

כלומר : L = 6, S = 4, W = 5, המימוש הנבחר הוא המימוש השני והמשתמש מעוניין בזמן הריצה של הפונקציה

<u>במקרה זה הפלט יהיה :</u>

No solution.
Function took 146 microseconds.

מכיוון שעבור קלט זה לא ניתן להגיע מהמצב ההתחלתי למצב הסופי (חשבו מדוע).
המספר 146 הוא כמובן רק לדוגמה.

<u>מימוש 1</u>

ממשו את האלגוריתם שנלמד בכיתה לפתרון בעיית הכדים.

1. בנו גרף מכוון ריק מקשתות המכיל את הקודקודים הרלוונטיים (חשבו כמה קודקודים צריך, ומה יהיו שמותיהם).
2. הוסיפו את הקשתות הנדרשות לגרף.
3. הריצו את אלגוריתם BFS על הגרף שהתקבל.

לצורך כך, הנכם <u>נדרשים</u> לממש את המחלקה הבאה:

גרף מכוון שממומש על ידי רשימות שכנות:

<u>פעולות בסיסיות:</u>

MakeEmptyGraph(n) – יצירת גרף ריק מקשתות עם n קודקודים

בפונקציות הבאות הקודקודים הם פרמטרים מסוג pair<int,int> כל אחד שמכיל את הערכים בכד הגדול (first) ובקטן (second)

GetAdjList(u) – החזרת רשימה מקושרת של השכנים של קודקוד u, ממוינת לקסיקוגרפית בסדר עולה כאשר, למשל, עבור השכנים הבאים, רשימה ממוינת תהיה (משמאל לימין):

<1,3> , <1,5> , <2,5>

AddEdge(u,v) – הוספת קשת (u,v)

<u>מימוש 2</u>

במימוש זה, נייעל את זמן הריצה בחלק מהמקרים, על ידי כך שלא נשמור ונקצה את כל קודקודי הגרף, אלא רק את הקודקודים והקשתות הרלוונטיים לחיפוש.

וביתר פירוט: לא נבנה את הגרף במלואו (לא נקצה $L \cdot S$ קודקודים ולא נאתחל את רשימת השכנויות), אלא רק קודקודים בהם אנו נתקלים כל עוד לא הגענו לפתרון.

נתחיל מהקודקוד המציין את המצב ההתחלתי. אם הגענו למצב הסופי, סיימנו, אין צורך לבנות את כל הגרף.

אחרת, נתקדם לשכנים ברמה הראשונה, אם אחד מהם מהווה פתרון - שוב סיימנו ואין צורך לחשב את כל הגרף.

ובאופן כללי, נקצה את הקודקודים שנמצאים ברמה ה- d-ית, רק אם לא מצאנו פתרון כשאר סרקנו את כל הקודקודים עד לרמה זו.

את רשימת השכנויות כלל לא נשמור בזכרון, אלא נממש פונקציה שבהנתן קודקוד מחשבת את שכניו.

על מנת לממש את האלגוריתם, יש לסמן באילו קודקודים כבר נתקלנו (כדי שלא נחזור לסרוק אותם שוב ושוב).

לצורך סימון הקודקודים ועל מנת שלא להקצות מקום שגודלו כמספר הקודקודים בגרף, יש להשתמש ב- **hashing**, לצורך מיפוי הקודקודים בהם כבר ביקרנו, כפי שלמדתם בסמסטר שעבר בקורס מבני נתונים (דוגמה לשימוש ב-hashing נמצאת בנספח א' בסוף מסמך זה).

<u>עליכם לממש את הפונקציות הבאות:</u>

א. פונקציה **CalculateAdjList(u)** – שבונה רשימה מקושרת של השכנים של קודקוד u שטרם נוצרו

ב. פונקציה המקבלת כקלט את L,S,W ופותרת את בעיית הכדים, תוך שימוש בהנחיות שלעיל.

בשני המימושים יש למדוד את זמן הריצה של הפונקציות, במידה והמשתמש בחר באפשרות זאת.

(דוגמה לשימוש ב-chrono נמצאת בנספח ב׳ בסוף מסמך זה).

בדיקת שגיאות:

הקפידו לבדוק שגיאות אפשריות בקלט. במקרה של שגיאה, כתבו הודעת שגיאה למסך *invalid input* וצאו מהתכנית באמצעות הפונקציה *exit(1)*.

(יש לבצע *#include <stdlib.h>* על מנת להשתמש בה).

הנחיות הגשה:

יש להגיש במערכת *mama* במקום המיועד להגשה את הקבצים הבאים:

1. קובץ *readme* שיכיל את כל פרטי ההגשה הבאים:

   כותרת – תרגיל תכנותי בקורס אלגוריתמים.

   שורה מתחת - שמות המגישים, מספרי ת.ז. שלהם ומספר הקבוצה של כל אחד מהם (מותר להגיש עם בן זוג מקבוצה אחרת).

   שימו לב: קובץ טקסט פשוט – לא *word*.

2. כל קבצי הקוד בסיומות *cpp*. ו- *h*.

רק אחד מבני הזוג יגיש את הפרויקט. הגשה כפולה תגרור קנס של 20 נק׳ לשני המגישים.

שימו לב! הגשה שאינה בפורמט הנדרש תידחה אוטומטית.

הערות הכרחיות נוספות: (שימו לב, למרות שזה קורס באלגוריתמים התכנית צריכה להיות כתובה לפי כל הכללים של תכנות נכון!)

- הקפידו על תיעוד, שמות בעלי משמעות למשתנים, מודולאריות וכל מה שנדרש מתכנות נכון. בתיעוד בראש התוכנית כתבו גם הוראות הפעלה מדויקות וברורות.
- הקפידו על חלוקה נכונה לקבצים (קבצי *cpp* וקבצי *h* לכל מחלקה).
- תכנתו *Object Oriented* והימנעו מאלמנטים מיותרים של קוד פרוצדוראלי.
- הקפידו לשחרר את כל הזיכרון אשר הקציתם דינמית לאחר שהשתמשתם בו.
- במקרה של תקלה בריצת התוכנית (מסיבה כלשהי), עליה לדווח זאת למשתמש טרם סיימה לרוץ.
- בדקו את תכניתכם על קלטים רבים ככל האפשר - כולל קלטים חוקיים ולא חוקיים, וזאת בנוסף לקלטים לדוגמה שהוכנו עבורכם במאמא.

בהצלחה!

צוות אלגוריתמים, סמסטר ב' תשפ"ה.

נספח א' – דוגמה לשימוש ב-hashing :

לפניכם תכנית קצרה שממחישה שימוש ב- hash. התכנית מתחזקת מאגר של מספרים שלמים, בעוד שבתכנית שלכם המאגר יהיה של אברים מסוג Vertex.

```
#include <iostream>
#include <unordered_set>
using namespace std;

int main() {
   // Create an unordered_set of integers
   unordered_set<int> unorderedSet;

   // Inserting elements
   unorderedSet.insert(10);
   unorderedSet.insert(5);
   unorderedSet.insert(20);
   unorderedSet.insert(15);



   // Checking if an element exists
   int searchElement;
   cout << "Enter an element\n";
   cin >> searchElement;

   if (unorderedSet.find(searchElement) != unorderedSet.end()) {
       cout << searchElement << " is presented in the set." <<  endl;
   }
   else {
       cout << searchElement << " is not presented in the set." <<  endl;
   }

   return 0;
}
```

נספח ב' – דוגמה לשימוש ב-chrono :

לפניכם תכנית קצרה שממחישה מדידת זמני ריצה של פונקציה.

```
#include <iostream>
#include <chrono>
#include <ios>
using namespace std;

void fun()
{
	/** the function **/
}

int main()
{
	auto start = chrono::high_resolution_clock::now();


	fun();// Here you put the name of the function you wish to measure

	auto end = chrono::high_resolution_clock::now();

	auto duration =
std::chrono::duration_cast<std::chrono::microseconds>(end - start);

	std::cout << "Function took " << duration.count() << "
microseconds." << std::endl;


	return 0;


}
```