פקולטה: מדעי הטבע
מחלקה: מדעי המחשב
שם הקורס: תכנות מערכות ב
קוד הקורס: 7023010-2 כל הקבוצות
מועד __**ב**___ סמסטר: ___**ב**___ שנה: ___ה'תשפ"א
תאריך הבחינה: ה' באב ה'תשפ"א, 14/07/2021
משך הבחינה: שעתיים וחצי – 150 דקות
המרצים: אראל סגל-הלוי, ערן קאופמן, חרות סטרמן.
המתרגלים: יבגני נייטרמן, אריה יטיב, אור אביטל.

פתרון הבחינה

אנא קיראו היטב את כל ההנחיות והשאלות לפני שתתחילו לכתוב.

יינתן מענק של 2 נקודות לסטודנטים שיכתבו את הפתרון באופן ברור קריא וקל לבדיקה, בפרט:

- השאלות פתורות במחברת הבחינה לפי הסדר שבשאלון;
- כל שאלה מתחילה בעמוד נפרד;
- הכתב ברור וקריא, בלי מחיקות קשקושים חיצים וטקסט מיותר.

אסור להשתמש בכל חומר עזר.

יש לענות על כל השאלות במחברת הבחינה.
אין צורך להעתיק את השאלון למחברת - השאלון יתפרסם לאחר הבחינה.

יש לענות תשובות מלאות, להסביר כל תשובה בפירוט, ולכתוב תיעוד לקוד.
יש לענות תשובות ממוקדות - לא יינתנו נקודות על תשובות עם טקסט מיותר שאינו קשור לנושא.

אם אתם לא זוכרים, לא בטוחים או לא מבינים משהו בשאלה כלשהי - נסו לפתור את השאלה כמיטב יכולתכם, הסבירו מה הבנתם ולמה התכוונתם, והמשיכו לשאלה הבאה. אל "תיתקעו" בשאלה אחת. מומלץ להשקיע עד 20 דקות בשאלה.

בהצלחה!!!

שאלה 0 [עד 64 נק']

[ציון על מטלות + מענקים]

שאלה 1 [9 נק'] - תור דו-כיווני

א [6 נק']. כתבו פונקציה בשם printKMax, המקבלת כקלט וקטור של מספרים ממשיים, ומספר שלם k, ומדפיסה את האיבר הגדול ביותר בכל תת-וקטור רציף באורך k. חתימת הפונקציה:

```
void printKMax(vector<double> input, int k) { ... }
```

תוכנית ראשית לדוגמה:

```
int main() {
     printKMax({11,22,33,44,33,22,11}, 3);
          // prints 33 44 44 44 33. Explanation:
          // The maximum of 11,22,33 is 33
          // The maximum of 22,33,44 is 44
          // The maximum of 33,44,33 is 44
          // The maximum of 44,33,22 is 44
          // The maximum of 33,22,11 is 33
     printKMax({55,44,33,22,11,22,33,44,55}, 4);
          // prints 55 44 33 33 44 55
}
```

שימו לב: **המימוש צריך להיות יעיל ולעבוד מהר גם על וקטורים ארוכים מאד וכש-k גדול מאד**. מומלץ להשתמש בתור דו-כיווני – deque. בנספח נמצא דף ההסבר על deque מאתר cppreference.

ב [3 נק']. הדגימו את פעולת הפונקציה שכתבתם על הקלט הראשון מהתוכנית הראשית (11,22,33,44,33,22,11), חלון בגודל 3. תארו את מצבם של מבני-הנתונים והמשתנים הפנימיים שהשתמשתם בהם בכל שלב.

פתרון: הפתרון האינטואיטיבי הוא לעבור בלולאה חיצונית על וקטור הקלט (נניח שהוא בגודל n), ולעבור בלולאה פנימית על בלוק בגודל k המתחיל מהאינדקס של הלולאה החיצונית, ולחשב את המקסימום בבלוק הזה. אבל הסיבוכיות של הפתרון הזה O(n*k) וזה עלול להיות לא יעיל כשהוקטור גדול (סטודנטים שפתרו את מטלת האקראנק באופן זה קיבלו timeout; פתרון זה זכה בעד 50% מהנקודות, בהתאם לרמת ההסבר).

פתרון קצת יותר יעיל הוא, לשמור במשתנה את המקסימום הנוכחי, ולא לחשב אותו בכל בלוק מחדש, אלא לחשב אותו רק אם יש סיכוי שהוא השתנה – רק אם האיבר שעכשיו יצא מה"חלון" שווה למקסימום הנוכחי. בדרך-כלל זה לא יקרה, ולכן נצטרך לחשב את המקסימום פחות מ-n פעמים. אבל, עדיין יהיו מקרים שבהם נצטרך לחשב את המקסימום בכל פעם מחדש, למשל כשוקטור הקלט מסודר בסדר יורד: 9 8 7 6 5 ....
במקרה זה, המקסימום תמיד ייצא מהחלון, ותמיד נצטרך לחשב אותו מחדש. **פתרון זה יזכה במלוא הנקודות אם הוא מוסבר היטב.**

הפתרון היעיל הוא, לשמור לא רק את המקסימום הנוכחי, אלא את כל האיברים *שיש להם אפשרות* להיות מקסימליים אחרי שהמקסימום הנוכחי ייצא מהחלון. אלה הם פשוט האיברים הגדולים יותר מאלה שבאים אחריהם. הם מחכים בתור עד שהמקסימום הנוכחי ייצא, ואז הם נעשים מקסימליים.

כדי שנוכל לדעת מתי המקסימום הנוכחי ייצא, צריך לשמור לא את האיברים עצמם אלא את האינדקסים שלהם; להשוות את האינדקס למונה של הלולאה, ואם הוא קטן יותר – להוציא אותו מהתור.

```
void printKMax(vector<double> input, int k) {
    deque<int> dq;
    for (int i=0; i<input.size(); ++i) {

        double new_item = input[i];
        while (!dq.empty()) {
            double waiting_item = input[dq.back()];
            if  (waiting_item <= new_item) {
                dq.pop_back();  // the waiting item has no chance to be the
                                // maximum, since it is smaller than the new item.
            } else {
                break;   // the waiting item still has a chance
                         // to be  maximum, so it keeps waiting in the queue.
            }
        }
        dq.push_back(i);  // the new item is now waiting in the queue.

        while (!dq.empty() && dq.front()+k <= i) {
            dq.pop_front();   // the front item is outside the window.
        }

        if (i >= k-1) {
            double current_max_item = input[dq.front()];
            cout << current_max_item << " ";
        }
    }
    cout << endl;
}
```

הנה דוגמה על הקלט מהתוכנית הראשית:

11,22,33,44,33,22,11 חלון בגודל 3.

איטרציה 0: מכניסים לתור את איבר 0 (= 11).

איטרציה 1: משוים בין סוף התור הנוכחי - איבר 0 (= 11) לאיבר 1 (= 22); כיוון שהשני גדול יותר, אין טעם לשמור את הראשון – הוא אף פעם לא יהיה מקסימום – אז מוציאים את 0 מהתור ומכניסים את 1

איטרציה 2: באותו אופן, סוף התור הנוכחי - איבר 1 (= 22) כבר לא יהיה מקסימום, אז מוציאים אותו מהתור ומכניסים את איבר 2 (= 33).

בשלב זה הגענו כבר לבלוק הראשון בגודל k, אז אנחנו יכולים להדפיס. מדפיסים את האיבר הראשון בתור: 33.

איטרציה 3: משוים בין סוף התור הנוכחי - איבר 2 (= 33) לאיבר 3 (= 44); כיוון שהשני גדול יותר, אין טעם לשמור את הראשון – הוא אף פעם לא יהיה מקסימום – אז מוציאים את 2 מהתור ומכניסים את 3. מדפיסים את האיבר בראש התור: 44.

איטרציה 4: משוים בין סוף התור הנוכחי - איבר 3 (= 44) – לאיבר 4 (=33); כיוון ש**הראשון** גדול יותר, **לא** מוציאים אותו מהתור, אלא מכניסים את איבר 4 אחריו בתור (איבר 4 מחכה בתור – כשאיבר 3 ייצא, הוא יכול להיות המקסימום). מדפיסים את ראש התור – 44.

איטרציה 5: משוים בין סוף התור הנוכחי - איבר 4 (=33) – לאיבר 5 (=22). כיוון ש**הראשון** גדול יותר, **לא** מוציאים אותו מהתור, אלא מכניסים את איבר 5 אחריו בתור. מדפיסים את ראש התור – 44.

עכשיו מוציאים את איבר 3 (= 44) מראש התור, כי הוא כבר לא בחלון (האינדקס שלו – 3 – קטן מדי).

איטרציה 6: משוים בין סוף התור הנוכחי - איבר 5 (=22) – לאיבר 6 (=11). כיוון ש**הראשון** גדול יותר, **לא** מוציאים אותו מהתור, אלא מכניסים את איבר 6 אחריו בתור. מדפיסים את ראש התור הנוכחי – שהוא 33 (אחרי שה-44 יצא).

שאלה 2 [9 נק'] - זיכרון

נתונה המחלקה הבאה:

```
1. class Vector {
2.     int m_size=0;                    // default initialization
3.     int* m_arr=new int [10];  // default initialization
4.     Vector(int size) { m_size=size; m_arr = new int[size]; } // custom
size initialization
5.         ~Vector(){ delete[] m_arr;};
6. };
```

א. המחלקה עלולה לגרום לזליגת זיכרון. הסבר מדוע, והסבר מה צריך לשנות בבנאי בשורה 4 כדי שלא תהיה זליגת זיכרון ממנו (יש לכתוב את הבנאי לאחר השינוי; אין לבצע שינויים נוספים).

ב. מה ראוי להוסיף למחלקה כדי שתעבוד כמו שצריך, בהתאם ל"כלל השלושה" (Rule of Three) שלמדנו בהרצאות? ענה בכתיבת הקוד הרלוונטי.

ג. הוספתי תוכנית main כדלקמן אך התוכנית לא מתקמפלת. תקן את השגיאות במחלקה (ייתכן יותר מאחת) כדי שהתוכנית תתקמפל.

```
int main() {
    Vector matrix[10];
}
```

הערה : ניתן לפתור את שלושת הסעיפים ע״י כתיבת כל המחלקה בבת אחת ובתוספת הערות

פתרון:

א [2 נק']. תהיה זליגת זכרון כי אנחנו קוראים תמיד לפקודה בשורה 3, ואז קוראים תמיד לבנאי המבצע הקצאה נוספת באותו מקום [1 נק']. הפתרון - להעביר את האתחול לרשימת אתחול. זה יבטיח שיתבצע רק איתחול אחד [1 נק']:

```
Vector(int size):m_arr(new int [size]),m_size(size)  {} ;
```

הערות:

- למחוק את ה-new מהבנאי זה לא פתרון טוב, כי אז הוקטור תמיד יהיה בגודל 10 בלי להתחשב בגודל שהעברנו לבנאי.
- להוסיף delete לבנאי – פתרון אפשרי, אבל פחות יעיל (ניתן ניקוד חלקי בהתאם לרמת ההסבר).

ב [4 נק']. "כלל השלושה" אומר, שמחלקה עם ניהול זיכרון דינמי צריכה את שלוש השיטות: מפרק, בנאי מעתיק, ואופרטור השמה, המבצעים העתקה עמוקה (וגם מי שלא זכר את המושג "כלל השלושה" יכל לזכור שמבנה-נתונים עם הקצאת-זיכרון על הערימה צריך העתקה עמוקה).

כיוון שכבר יש מפרק במחלקה, צריך להוסיף גם את שני האחרים [לכל אחד 2 נק']:

```
Vector& operator=(const Vector& other) {
	if (this != &other) {
		m_size=other.m_size;
		delete[] m_arr;
		m_arr = new int[other.m_size];
		for (int i=0; i< other.m_size;i++)
			m_arr[i] = other.m_arr[i];
	}
	return *this;
};
Vector(const Vector& other) {
	m_arr = new int[other.m_size];
	for (int i=0; i< other.m_size;i++)
		m_arr[i] = other.m_arr[i];
};
```

ג [3 נק']. חסר בנאי ריק – דיפולטיבי [1.5 נק'], והבנאים אינם ציבוריים [1.5 נק']. הפתרון המלא הוא:

```
class Vector {
	int* m_arr=new int [10];
	int m_size=0;
public:
	Vector()=default;
	Vector(int size):m_arr(new int [size]),m_size(size)  {};
	Vector& operator=(const Vector& other) {
		m_size=other.m_size;
		m_arr = new int[other.m_size];
		for (int i=0; i< other.m_size;i++)
			m_arr[i] = other.m_arr[i];
```

```
        return *this;
    };
    Vector(const Vector& other) {
        *this=other;
    };
    ~Vector(){ delete[] m_arr;};
};
```

הערה: אין צורך להגדיר אופרטור סוגריים מרובעים. הסוגריים המרובעים בתוכנית הראשית מגדירים מערך של 10 עצמים מסוג וקטור – ולא קוראים לאופרטור סוגריים מרובעים של הוקטור עצמו. למדנו על זה בשיעור על בנאים.

שאלה 3 [9 נק'] - מספר עם יחידות

הוסיפו למחלקה NumberWithUnits, שכתבתם במטלה 3, **אופרטור חילוק (/)**. האופרטור צריך לוודא שלשני הארגומנטים שלו יש יחידות תואמות (כגון: שניהם אורך, שניהם משקל וכו' – לפי קובץ היחידות). אם היחידות לא תואמות – יש לזרוק חריגה. אם היחידות תואמות – יש להחזיר את היחס ביניהן **כמספר ממשי**.
מצורפת תוכנית ראשית לדוגמה.

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

#include "NumberWithUnits.hpp"
using namespace ariel;

int main() {
      ifstream units_file{"units.txt"};
      NumberWithUnits::read_units(units_file);

      NumberWithUnits a{2, "km"}; // 2 kilometers
      NumberWithUnits b{500, "m"}; // 500 meters
      cout << (a/b) << endl; // prints 4, since 2 km = 4 * 500 m.
      cout << (b/a) << endl; // prints 0.25, since 500 m = 0.25 * 2 km.

      NumberWithUnits c{2, "kg"}; // 2 kilograms
      cout << (b/c) << endl; // throws an exception, e.g.
         //"units [km] and [kg] do not match."
      return 0;
}
```

א. כתבו את הקוד שצריך להוסיף לקובץ הכותרת NumberWithUnits.hpp.

פתרון [3 נק']. אפשר לממש את האופרטור בגוף המחלקה או כפונקציה חברה. במקרה השני יש לכתוב:

```
friend double operator/(const NumberWithUnits &a, const NumberWithUnits &b);
```

ב. כתבו את הקוד שצריך להוסיף לקובץ המימוש NumberWithUnits.cpp. מותר לכם להשתמש בפונקציות שמימשתם במטלה עצמה – אין צורך לממש אותן מחדש – אבל יש להסביר בפירוט מה הפונקציות האלו עושות.

פתרון [6 נק']. יש הרבה אפשרויות, תלוי איך בדיוק פתרתם את המטלה. למשל, אם היתה לכם שיטה סטטית בשם convert, שכבר זורקת חריגה אם היחידות לא תואמות, אז אפשר לממש את האופרטור החדש כך:

```
double operator/(const NumberWithUnits &a, const NumberWithUnits &b) {
      double converted_b_value =
            NumberWithUnits::convert(b.value, a.unit, b.unit);
      return a.value / converted_b_value;
}
```

שאלה 4 [6 נק'] – לינוקס

נתונה תיקיה במערכת לינוקס, ובה הקבצים הבאים:

```
drwxr-xr-x 2    4096 Jul  3 22:39 .
drwxrwxr-x 3    4096 Jul  3 22:27 ..
-rw-rw-r-- 1      91 Jul  3 22:39 hello.cpp
-rw-rw-r-- 1      39 Jul  3 22:39 run.sh
-rw-rw-r-- 1      39 Jul  3 22:39 string.txt
```

תוכן הקבצים:

hello.cpp:

```
#include <iostream>
using namespace std;
int main() {
      cout << "Enter a string: ";
      string s;
      cin >> s;
      cout << endl << "The string you entered is " << s << endl;
}
```

run.sh:

```
clang++-9 -std=c++2a $1
./a.out
```

string.txt:

```
abcdefghij
```

כשמריצים בשורת הפקודה:

```
$ ./run.sh hello.cpp
```

מתקבלת הודעת שגיאה:

```
bash: ./run.sh: Permission denied
```

א [2 נק']. הסבירו את הודעת השגיאה וממה היא נובעת.

פתרון: השגיאה נובעת מכך שלקובץ `run.sh` אין הרשאות הרצה – אפשר לראות שבשורה שלו ברשימת הקבצים אין "x".

ב [2 נק']. הסבירו איך לתקן את השגיאה - מה צריך לעשות כדי שאותה שורה בדיוק "**run.sh hello.cpp/.**" תקמפל את התוכנית הנמצאת בקובץ `hello.cpp` ותריץ אותה. הסבירו מה צריך לשנות בקבצים – אם צריך, ואיזה פקודות לינוקס צריך לבצע – אם צריך. **יש לכתוב בפירוט את כל הקוד הדרוש.**

פתרון: לא צריך לשנות שום דבר בקבצים, צריך רק לשנות את ההרשאות, למשל בפקודה:

```
chmod +x run.sh
```

או בפקודה:

```
chmod 755 run.sh
```

ג [2 נק']. אחרי התיקון של סעיף ב, כשמריצים "**`./run.sh hello.cpp`**", התוכנית עוצרת ומחכה לקלט מהמשתמש. כתבו פקודה ב-bash שתריץ את התוכנית כך שלא תחכה לקלט מהמשתמש, אלא תקבל קלט מהקובץ string.txt. למשל, כשמריצים עם תוכן הקובץ הנוכחי, יתקבל הפלט:

```
The string you entered is abcdefghij
```

פתרון: צריך להשתמש באופרטור הכוונת קלט:

```
./run.sh hello.cpp < string.txt
```

שאלה 5 [9 נק'] – פרמוטציות

בקורס "תיכנות מערכות ג++", סכום ציוני המטלות בקורס נקבע באופן הבא: עוברים על המטלות לפי הסדר ומחברים את הציונים שלהן, עד שהסכום מגיע ל-48 או יותר; המטלות הבאות לא נחשבות. לדוגמה, אם ציוני המטלות הם:

```
10  20  30  40  50
```

אז הסכום יהיה 30+20+10 = 60.

לעומת זאת, אם ציוני המטלות הם:

```
40  50  30  20  10
```

אז הסכום יהיה 40 + 50 = 90.

סטודנטים רבים התלוננו שזה לא הוגן, ששני אנשים קיבלו אותם ציונים בדיוק רק בסדר אחר, והציון הסופי שלהם יהיה שונה. לכן הוחלט לאפשר לכל סטודנט לסדר את המטלות שלו כרצונו. עליכם לכתוב פונקציה שתעזור לסטודנטים לסדר את המטלות באופן שייתן להם את הציון הגבוה ביותר האפשרי ע"פ הכללים. הפונקציה צריכה לעבור על כל הדרכים האפשריות לסדר את המטלות, לחשב את הסכום בכל סדר אפשרי, ולהחזיר את הסכום הגבוה ביותר.

העזרו בפונקציות הבאות מהספריה התקנית:

```
template< class RandomIt >
    void sort( RandomIt first, RandomIt last );

template< class BidirIt >
    bool next_permutation( BidirIt first, BidirIt last );
```

שימו לב: בנספח תוכלו למצוא את הדף המלא של הפונקציה `next_permutation` מתוך אתר `cppreference`.

כותרת הפונקציה שאתם צריכים לכתוב:

```
int best_sum(vector<int> grades) {..}
```

תוכנית ראשית לדוגמה:

```
int main() {
      cout << best_sum({10,20,30,40,50}) << endl; // 90
      cout << best_sum({50,40,30,20,10}) << endl; // 90
      cout << best_sum({40,50,30,20,10}) << endl; // 90
      cout << best_sum({10,10,32,10,10,33,10,10}) << endl; // 75=32+33+10
}
```

פתרון: בתחילת הפונקציה יש לסדר את הוקטור בסדר עולה [2 נק'], ואחר-כך לעבור על כל הפרמוטציות כמו שלמדנו בהרצאה על אלגוריתמים בספריה התקנית [5 נק'].

כדי לחשב את הציון בכל פרמוטציה בנפרד, נכתוב פונקציית עזר [2 נק'] (לא חובה – אפשר גם לשים את המימוש בלולאה הראשית):

```
#include <algorithm>
#include <numeric>
#include <iostream>
using namespace std;

const int STOP_GRADE = 48;
int calculate_sum(vector<int> grades) {
	int sum = 0;
	for (int g: grades) {
		sum += g;
		if (sum >= STOP_GRADE) { break; }
	}
	return sum;
}
```

עכשיו נעבור על כל הפרמוטציות לפי הסדר:

```
int best_sum(vector<int> grades) {
	int best_sum_so_far = 0;
	sort(grades.begin(), grades.end());
	do {
		int sum = calculate_sum(grades);
		if (sum > best_sum_so_far) {
			best_sum_so_far = sum;
		}
	} while (next_permutation(grades.begin(), grades.end()));
	return best_sum_so_far;
}
```

שאלה 6 [8 נק'] – ירושה

נתון הקוד הבא:

```
1 . #include <iostream>
2 . using namespace std;
3 .
4 . class A
5 . {
6 . public:
7 .     virtual void f(const int i) { cout << "A::f" << endl; }
8 .     void g(const int i) { cout << "A::g" << endl; }
9 .     void h(const int i) { cout << "A::h" << endl; }
10.     virtual void i(const int i) { cout << "A::i" << endl; }
11. };
12.
13. class B : public A
14. {
15. public:
16.     void f() { cout << "B::f" << endl; }
17.     void g() { cout << "B::g" << endl; }
18. };
19.
20. class C : public B
21. {
22. public:
23.     void f(const int i) { cout << "C::f" << endl; }
24.     void i(const int i) { cout << "C::i" << endl; }
25. };
26.
27. class D : public C
28. {
29. public:
30.     void f(const int i) { cout << "D::f" << endl; }
31.     void g(const int i) { cout << "D::g" << endl; }
32.     void h(const int i) { cout << "D::h" << endl; }
33.     void i() { cout << "D::i" << endl; }
34. };
35.
36. int main()
37. {
38.     D d;
39.     A* pa = &d;
40.     B* pb = &d;
41.     C* pc = &d;
42.     D* pd = &d;
43.
44.     pa->f(1);
45.     pb->f(1);
46.     pc->f(1);
```

```
47.     pc->f();
48.     pd->f(1);
49.
50.     pa->g(1);
51.     pb->g(1);
52.     pc->g(1);
53.     pd->g(1);
54.
55.     pa->h(1);
56.     pb->h(1);
57.     pc->h(1);
58.     pd->h(1);
59.
60.     pa->i(1);
61.     pd->i(1);
62.
63.     A a, *p = &a;
64.     p->f(1);
65. }
```

עבור שורות 44-64 (לא כולל שורה 63 ולא כולל שורות ריקות) יש לכתוב האם השורה מתקמפלת או לא.
אם השורה מתקמפלת, יש לכתוב מה הפלט של השורה.

פתרון [כל שורה = חצי נקודה]:

```
44.     pa->f(1); // D::f
45.     pb->f(1); // הפונקציה מוסתרת B::f. לא מתקמפל. אין פרמטרים לפונקציה.
46.     pc->f(1); // D::f
47.     pc->f(); // הפונקציה מוסתרת C::f. לא מתקמפל. יש פרמטר לפונקציה
48.     pd->f(1); // D::f
49.
50.     pa->g(1); // A::g
51.     pb->g(1); // הפונקציה מוסתרת B::g. לא מתקמפל. אין פרמטרים לפונקציה
52.     pc->g(1); // הפונקציה מוסתרת B::g. לא מתקמפל. אין פרמטרים לפונקציה
53.     pd->g(1); // D::g
54.
55.     pa->h(1); // A::h
56.     pb->h(1); // A::h
57.     pc->h(1); // A::h
58.     pd->h(1); // D::h
59.
60.     pa->i(1); // C::i
61.     pd->i(1); // הפונקציה מוסתרת D::i. לא מתקמפל. אין פרמטרים לפונקציה
62.
63.     A a, *p = &a;
64.     p->f(1); // A::f
```