

פקולטה: מדעי הטבע
מחלקה: מדעי המחשב
שם הקורס: תכנות מערכות ב
קוד הקורס: 2-7023010 כל הקבוצות
מועד ____ א ____ סמסטר: ____ ב ____ שנה: ____ ה'תש"ף
תאריך הבחינה: ט' בתמוז ה'תש"ף, 01/07/2020
משך הבחינה: 3 שעות
שם המרצים: אראל סגל-הלוי, ערן קאופמן
פתרון הבחינה

שאלה 1 [10 נק'] - מתוך האקראנק ++C

תקציר בעברית: בשאלה זו עליכם לתקן באג בקוד הנתון. הקוד אמור לחשב את הרווח של בית-מלון מהשכרת חדרים ודירות. המחירים נתונים ע"י הנוסחה:

- חדר רגיל (standard room): מספר חדרי שינה כפול 50 ועוד מספר אמבטיות כפול 100.
- דירה (apartment): מספר חדרי שינה כפול 50 ועוד מספר אמבטיות כפול 100 ועוד 100.

הרווח של בית-המלון הוא סכום הרווח שלו מכל החדרים שהוא משכיר.

בשורה הראשונה של הקלט נמצא מספר החדרים/דירות שהמלון משכיר. בשורות הבאות, כל שורה מתארת חדר או דירה, מספר חדרי-השינה, ומספר האמבטיות.

לדוגמה, אם הקלט הוא:

```
2
standard 3 1
apartment 1 1
```

אז הפלט הצפוי הוא:

500

כי המלון מרויח 250 מהחדר הרגיל שיש בו 3 חדרי-שינה + 1 אמבטיה, ועוד 250 מהדירה שיש בה 1 חדר-שינה + 1 אמבטיה.

אבל יש באג בתוכנית הגורם לשגיאה לוגית – הפלט המתקבל נמוך מדי – בדוגמה למעלה מתקבל הפלט 400.

א. [5 נק'] הסבירו ממה בדיוק נובעת השגיאה (ציינו מספרי שורות רלבנטיות).

ב. [5 נק'] הסבירו איך לתקן את השגיאה (ציינו מספרי שורות); אין לשנות את התוכנית הראשית.

בהמשך מובא הסבר מלא באנגלית; אם הבנתם את ההסבר בעברית אתם יכולים לדלג עליו ולעבור ישיר לקוד.

Hotel Prices

In this challenge, the task is to debug the existing code to successfully execute all provided test files.

The given code defines two classes `HotelRoom` and `HotelApartment` denoting respectively a standard hotel room and a hotel apartment. An instance of any of these classes has two parameters: `bedrooms` and `bathrooms` denoting respectively the number of bedrooms and the number of bathrooms in the room. The prices of a standard hotel room and hotel apartment are given as:

- Hotel Room: $50 * \text{bedrooms} + 100 * \text{bathrooms}$
- Hotel Apartment: The price of a standard room with the same number bedrooms and bathrooms plus 100.

For example, if a standard room costs 200, then an apartment with the same number of bedrooms and bathrooms costs 300.

In hotel's codebase, there is a piece of code reading the list of rooms booked for today and calculates the total profit for the hotel. However, sometimes calculated profits are lower than they should be.

Debug the existing `HotelRoom` and `HotelApartment` classes' implementations so that the existing code computing the total profit returns a correct profit.

Your function will be tested against several cases by the locked template code.

Input Format

The input is read by the provided locked code template.

In the first line, there is a single integer n denoting the number of rooms booked for today.

After that n lines follow. Each of these lines begins with a `room_type` which is either `standard` or `apartment`, and is followed by the number of bedrooms and the number of bathrooms in this room.

Constraints

- $1 \leq n \leq 100$.
- There is at least 1 and at most 5 bedrooms in a room.
- There is at least 1 and at most 5 bathrooms in a room.

Output Format

The output is produced by the provided and locked code template. It calculates the total profit by iterating through the vector of all rooms read from the input.

Sample Input 0

```
2
standard 3 1
apartment 1 1
```

Sample Output 0

500

Explanation 0

In the sample we have one standard room with 3 bedrooms and 1 bathroom, and one apartment with 1 bedrooms and 1 bathroom. The price for the room is $3 \times 50 + 100 = 250$. The price for the apartment is $50 + 100 + 100 = 250$. Thus the hotel profit is as the sum of prices of both rooms.

Code

```
1  #include <iostream>
2  #include <vector>
3
4  using namespace std;
5
6  class HotelRoom {
7  public:
8      HotelRoom(int bedrooms, int bathrooms)
9          : bedrooms_(bedrooms), bathrooms_(bathrooms) {}
10
11     int get_price() {
12         return 50*bedrooms_ + 100*bathrooms_;
13     }
14 private:
15     int bedrooms_;
16     int bathrooms_;
17 };
18
19 class HotelApartment : public HotelRoom {
20 public:
21     HotelApartment(int bedrooms, int bathrooms)
22         : HotelRoom(bedrooms, bathrooms) {}
23
24     int get_price() {
25         return HotelRoom::get_price() + 100;
26     }
27 };
28
29 int main() {
30     int n;
31     cin >> n;
32     vector<HotelRoom*> rooms;
33     for (int I = 0; I < n; ++i) {
34         string room_type;
35         int bedrooms;
36         int bathrooms;
37         cin >> room_type >> bedrooms >> bathrooms;
38         if (room_type == "standard") {
39             rooms.push_back(new HotelRoom(bedrooms, bathrooms));
40         } else {
41             rooms.push_back(new HotelApartment(bedrooms, bathrooms));
42         }
43     }
44
45     int total_profit = 0;
46     for (auto room : rooms) {
47         total_profit += room->get_price();
48     }
49 }
```

```

47     }
48     cout << total_profit << endl;
49
50     for (auto room : rooms) {
51         delete room;
52     }
53     rooms.clear();
54
55     return 0;
56 }

```

פתרון (ע"פ הרצאה 6 ומטלת האקראנק):

א. השגיאה הלוגית נובעת מכך שהשיטה `get_price` במחלקה `HotelRoom` לא וירטואלית. כתוצאה מכך בשורה 46 קוראים תמיד ל `get_price` של המחלקה `HotelRoom` גם כשהעצם הוא מסוג `HotelApartment`.

ב. יש להוסיף `virtual` בשורה 11:

```

11     virtual int get_price() {

```

הערה:

- אין צורך להוסיף מפרק למחלקה, כי אין בה שום דבר מיוחד שצריך לפרק (כגון זיכרון דינמי).

שאלה 2 [10 נק'] - מתוך האקראנק לינוקס

תקציר בעברית: כתבו פקודה בלינוקס, המקבלת שורות טקסט דרך הקלט התקני, ומחליפה כל הופעה של המילה "thy" במילה "your".

- אין להבחין בין אותיות גדולות לקטנות – יש להחליף גם `THY` וגם `tHy` וכו'.
- יש להחליף מילים שלמות בלבד – למשל את המחרוזת שבתוך `xxxTHYzzz` אין להחליף כלל.

Task

For each line in a given input file, transform all the occurrences of the word 'thy' with 'your'. The search should be **case insensitive**, i.e. 'thy', 'Thy', 'tHy' etc. Should be transformed to 'your'.

Input Format

A text file will be piped into your command via STDIN.

Output Format

Transform and display the text as required in the task.

Sample Input

The line:

"Feed'st thy light's flame with self-substanthyal fuel,"

should be transformed to:

"Feed'st **your** light's flame with self-substan**thy**al fuel,"

The line:

"**Thy** self **thy** foe, to **thy** sweet self too cruel:"

should be transformed to:

"**your** self **your** foe, to **your** sweet self too cruel:"

פתרון (ע"פ תירגול 2 ומטלת האקראנק):

הפקודה המתאימה ביותר להחלפת טקסטים היא sed. יש כמה פתרונות, הנה שניים:

```
sed 's/\b[tT][hH][yY]\b/your/g'
sed 's/\bthy\b/your/gi'
```

מפתח ניקוד זמני: 4 - sed, גבולות מילה - 2, אותיות גדולות וקטנות - 2, החלפת כל המילים - 2.

הערה: הפתרון של החלפת רווחים לפני ואחרי, כמו כאן:

```
sed 's/ thy / your /gi'
```

מפספס מילים בתחילת השורה ובסופה.

שאלה 3 [10 נק']

נתונה מחלקה Board המייצגת לוח במשחק איקס-עיגול, וכן תוכנית ראשית להדגמה:

```
01 #include <iostream>
02 #include <string>
03 #include <cmath>
04 #include <vector>
05 #include <utility>
06
07 using namespace std;
08
09 class Board {
10     vector<vector<char>> board;
11 public:
12     Board(): board(3, vector<char>(3, '.')) {}
13
14     char operator[](pair<int,int> rowcol) const {
```

```

15         return board[rowcol.first][rowcol.second];
16     }
17
18     char& operator[](pair<int,int> rowcol) {
19         return board[rowcol.first][rowcol.second];
20     }
21 };
22
23 int main() {
24     Board b;
25     cout << b[{1,1}] << endl; // should print "."
26     b[{1,1}] = 'X';
27     cout << b[{1,1}] << endl; // should print "X"
28     try {
29         b[{1,1}] = 'R'; // should throw an exception
30     } catch (string message) {
31         cout << message << endl; // should print "illegal char"
32     }
33     cout << b[{1,1}] << endl; // should print "X"
34 }

```

התוכנית רצה ועובדת, אבל היא לא זורקת שום חריגה; אנחנו רוצים שתזרוק חריגה כשמישהו מנסה להכניס ללוח תו שהוא לא איקס ולא עיגול (למשל בשורה 29). עליכם לשנות את הקוד כך שתזרוק חריגה במקרים אלה, מבלי לשנות את התוכנית הראשית.

רמז: יש לשנות את שורה 18 כך שהמבנה Board לאחר השינוי ייראה כך:

```

09 class Board {
10     vector<vector<char>> board;
11 public:
12     Board(): board(3, vector<char>(3, '.')) {}
13
14     char operator[](pair<int,int> rowcol) const {
15         return board[rowcol.first][rowcol.second];
16     }
17
18     char_reference operator[](pair<int,int> rowcol) {
19         return board[rowcol.first][rowcol.second];
20     }
21 };

```

מה שנשאר לכם לעשות זה רק לכתוב את המחלקה char_reference.

פתרון (ע"פ הרצאה 9 ומטלה 4):

```

class char_reference {
    char& cr;
public:
    char_reference(char& cr): cr(cr) {}

```

```

char_reference& operator=(char c) {
    if (c=='X' || c=='O') {
        cr = c;
    } else {
        throw string("illegal char");
    }
    return *this;
}

operator char() {
    return cr;
}
};

```

מפתח ניקוד זמני: שדות – 2, בנאי – 2, אופרטור השמה – 4, אופרטור המרה – 2.

הערות:

- אופרטור המרה דרוש כדי שהתוכנית הראשית תוכל להדפיס את האות בשורות 25, 27.
- במקום אופרטור המרה אפשר לשים גם אופרטור פלט.
- סטודנטים שאלו האם לא צריך לשנות גם את שורה 19 ולהמיר ל char_reference? תשובה: אפשר אבל לא חובה – ההמרה מתבצעת אוטומטית ע"י קריאה לבנאי הממיר.
- סטודנטים שפתרו ע"י כתיבת אופרטור = עם שני פרמטרים לא קיבלו נקודות – זה לא יעבוד – אופרטור = עובד רק עם פרמטר אחד (בנוסף לפרמטר this). נסו ותראו
- סטודנטים שפתרו ע"י זריקת חריגה מתוך אופרטור [] לא קיבלו נקודות – גם זה לא יעבוד – כי תיזרק חריגה גם בקריאה ולא רק בכתיבה.
- מאותה סיבה, זריקת חריגה מתוך הבנאי של char_reference לא תעבוד.
- המחלקה char_reference לא צריכה לרשת מ Board ולא צריכה להכיל שדה מסוג Board או מסוג וקטור, אלא רק שדה רפרנס.
- חשוב שהשדה במחלקה יהיה &char ולא רק char, כדי שההשמה תעבוד.

שאלה 4 [10 נק']

כיתבו תוכנית המקבלת קובץ קלט בשם input.txt המכיל מספרים שלמים חיוביים או שליליים (כל מספר בשורה אחת), וכותבת שני קבצי פלט: positive.txt, negative.txt. הקובץ positive.txt אמור להכיל את כל המספרים החיוביים בקלט, כולל אפס; הקובץ negative.txt אמור להכיל את כל המספרים השליליים בקלט, לא כולל אפס. לדוגמה, אם קובץ הקלט הוא:

Input.txt:

```

-1
4
55
0
-8
9

```

אז קבצי הפלט הם:

positive.txt:

4
55
0
9

negative.txt:

-1
-8

כבר כתבנו עבורכם את התוכנית הראשית – אתם צריכים רק להשלים את החסר בשורה האחרונה (המודגשת). אין לבצע שינויים נוספים בקוד.

```
#include <iostream>
#include <fstream>
#include <iterator>
#include <algorithm>
using namespace std;

int main() { // a demo program
    ifstream input("input.txt");
    ofstream positive("positive.txt");
    ofstream negative("negative.txt");
    partition_copy(...);    // COMPLETE THIS LINE
}
```

שימו לב: בנספח לבחינה ישנו דף מהאתר של שפת ++C המסביר על הפונקציה partition_copy. מומלץ להיעזר בו.

פתרון (ע"פ הרצאות 11-12):

כמו שרואים בתיעוד, הפונקציה partition_copy מצפה לקבל איטרטורים של קלט ופלט. צריך להעביר לה איטרטור קלט על input.txt ואיטרטור פלט על positive.txt, negative.txt. לשם כך יש להשתמש ב istream_iterator, ostream_iterator. הפרמטר האחרון הוא פרידקט (פונקטור):

```
partition_copy(
    istream_iterator<int>(input),
    istream_iterator<int>{},
    ostream_iterator<int>(positive, "\n"),
    ostream_iterator<int>(negative, "\n"),
    [](int i) {return i>=0;}
);
```

מפתח ניקוד זמני: מספר פרמטרים תקין – 1, איטרטור קלט – 3, איטרטור פלט – 3, פונקטור – 3.

הערה:

- לפי ההנחיות היה צריך רק להשלים את השורה ולא להוסיף עוד קוד, ולכן היה צריך להשתמש בביטוי למדא.
- סטודנטים שהוסיפו פונקציית-עזר ומימשו אותה, הפסידו נקודה אחת; סטודנטים שהוסיפו פונקציית עזר ולא מימשו אותה הפסידו 3 נקודות.
- סטודנטים שלא השתמשו ב `istream_iterator`, `ostream_iterator` כמו שלמדנו בהרצאה, הפסידו 2 נקודות.

שאלה 5 [10 נק']

נתונה התוכנית הבאה:

```
01  #include <iostream>
02  using namespace std;
03
04  class Complex {
05  private:
06      double _re;
07      double _im;
08  public:
09      Complex (double re= 0.0, double im= 0.0): _re(re), _im(im) {    }
10
11      const Complex operator-() const {
12          return Complex(-_re , -_im);
13      }
14
15      const Complex operator+(const Complex& other) const {
16          return Complex(_re + other._re, _im + other._im);
17      }
18
19      Complex& operator+=(const Complex& other) {
20          _re+= other._re;
21          _im+= other._im;
22          return *this;
23      }
24  };
25
26  ostream& operator<< (ostream& os, const Complex& c) {
27      return (os << c._re << '+' << c._im << 'i');
28  }
29
30  int main() {
31      Complex c1(1,2);
32      Complex c2(2,3);
33      cout << c1 + c2 << endl;  // should print "3+5i"
34      return 0;
35  }
```

כשמריצים אותה מתקבלת שגיאת קומפילציה:

```
ComplexDemo.cpp:27:21: error: '_re' is a private member of 'Complex'
    return (os << c._re << '+' << c._im << 'I');
                   ^
```

```
ComplexDemo.cpp:06:12: note: declared private here
    double _re;
           ^
```

```
ComplexDemo.cpp:27:37: error: '_im' is a private member of 'Complex'
```

```
return (os << c._re << '+' << c._im << 'I');
      ^
```

ComplexDemo.cpp:07:12: note: declared private here
double _im;
^

א [5 נק']. מהי הדרך הנכונה ביותר (מבחינת הנדסת תוכנה) לתקן את השגיאה? אין לשנות את התוכנית הראשית.

ב [5 נק']. סטודנט ניסה לתקן את השגיאה מהסעיף הקודם ע"י העברת השורות 26-28 לפני שורה 24, כך שהתקבל הקוד הבא (ללא שינוי בתוכנית הראשית):

```
04 class Complex {
05 private:
06     double _re;
07     double _im;
08 public:
09     Complex (double re= 0.0, double im= 0.0): _re(re), _im(im) {      }
10
11     const Complex operator-() const {
12         return Complex(-_re , -_im);
13     }
14
15     const Complex operator+(const Complex& other) const {
16         return Complex(_re + other._re, _im + other._im);
17     }
18
19     Complex& operator+=(const Complex& other) {
20         _re+= other._re;
21         _im+= other._im;
22         return *this;
23     }
24
25     ostream& operator<< (ostream& os, const Complex& c) {
26         return (os << c._re << '+' << c._im << 'I');
27     }
28 };
```

הפעם התקבלה השגיאה הבאה:

ComplexDemo.cpp:25:14: error: overloaded 'operator<<' must be a binary operator (has 3 parameters)
ostream& operator<< (ostream& os, const Complex& c) {
^

מה סוג השגיאה וממה בדיוק היא נובעת? פרטו והסבירו את הודעת השגיאה.

פתרון (ע"פ הרצאה 4):

א. כדי לאפשר לאופרטור הפלט לגשת למשתנים הפרטיים של המחלקה, יש להפוך אותו ל"חבר" ע"י הוספת השורה הבאה בגוף המחלקה, למשל בין שורה 23 לשורה 24:

```
friend ostream& operator<< (ostream& os, const Complex& c);
```

הערות:

- מי שהסביר את הפתרון באופן כללי אבל לא כתב מימוש – קיבל 4 מתוך 5.
- מי שכתב שמספיק להוסיף את המילה *friend* בתחילת הפונקציה – קיבל 3 מתוך 5 (זה לא מספיק – צריך להוסיף שורה עם המילה *friend* בגוף המחלקה).
- לעומת זאת, מי שפתר את הבעיה ע"י שינוי *private* ל *public* – קיבל 1 מתוך 5, כי זה פותר בעיה אחת אבל יוצר בעיה גרועה יותר – פגיעה בהסתרת המידע. מהנדס תוכנה צריך לדעת לפתור בעיות בלי ליצור בעיות חדשות...
- פתרון חלופי הוא להוסיף *getter*, לשנות את האופרטור כך שישתמש בהם. מי שמימש את הפתרון הזה באופן מלא (כולל כל השינויים הדרושים בכל הקוד) קיבל את מלוא הנקודות.

ב. שגיאת קומפילציה. כשאנחנו מכניסים את האופרטור לתוך המחלקה, הוא אוטומטית מקבל עוד פרמטר – המשתנה *this* המייצג את העצם הנוכחי (כמו אופרטור החיבור שנמצא באותה מחלקה). אנחנו למעשה מנסים להגדיר אופרטור << עם שלושה פרמטרים, אבל האופרטור הזה מקבל רק שני פרמטרים.

שאלה 6 [10 נק']

נתונה התוכנית הבאה:

```
01  #include <iostream>
02  #include <fstream>
03  using namespace std;
04
05  struct RGB {
06      uint8_t red, green, blue;
07  public:
08      RGB(): red(0), green(0), blue(0) {}
09      RGB(uint8_t r, uint8_t g, uint8_t b): red(r), green(g), blue(b) {}
10  };
11
12  int main() {
13      const int dimx = 800, dimy = 800;
14      ofstream imageFile("image.ppm", ios::out | ios::binary);
15      imageFile << "P6" << endl << dimx << " " << dimy << endl << 255 << endl;
16      RGB* image[dimx*dimy];
17      for (int j = 0; j < dimy; ++j) { // row
18          for (int i = 0; i < dimx; ++i) { // column
19              image[dimx*j+i] = new RGB{255,0,0};
20          }
21      }
22
23      imageFile.write(reinterpret_cast<char*>(&image), 3*dimx*dimy);
24      imageFile.close();
25
26      for (int j = 0; j < dimy; ++j) { // row
27          for (int i = 0; i < dimx; ++i) { // column
28              delete image[dimx*j+i];
29          }
30      }
31
32      return 0;
33  }
```

התוכנית אמורה לצייר ציור מסויים ולשים אותו בקובץ `image.ppm`, אבל בפועל כשמריצים את התוכנית ופותחים את הקובץ – מגלים שם ציור אחר לגמרי:

א [5 נק']. מה הציור שאמור להיות בקובץ? הסבירו בפירוט איך בדיוק הציור אמור להגיע לקובץ (ציינו מספרי שורות רלבנטיות).

ב [5 נק']. מדוע הציור בפועל הוא שונה? מה סוג השגיאה וממה בדיוק היא נובעת? פרטו (ציינו מספרי שורות).

פתרון (ע"פ הרצאה 7):

א. הציור אמור להיות ריבוע אדום בגודל 800 על 800. בשורות 17-21 אנחנו יוצרים מערך של מבנים מסוג RGB, כל אחד מהם מייצג פיקסל. כל פיקסל מיוצג ע"י שלושה בתים רצופים – אדום, ירוק, כחול. במקרה זה אנחנו שמים ערך חיובי רק בפרמטר הראשון ולכן הכל אדום. בשורה 23 מבצעים המרה מסוג reinterpret שלמעשה לא משנה שום דבר בזיכרון – רק מאפשרת להעביר את המערך כפי שהוא ל write. הפונקציה write כותבת את הבתים מהזיכרון כמו שהם לקובץ הבינארי.

ב. זו שגיאה לוגית. הציור בפועל שונה כי אנחנו שמים במערך פוינטרים ל-RGB במקום RGB ממש. לכן כשאנחנו מבצעים העתקה בינארית בשורה 23, מה שמגיע לקובץ זה הכתובות שבתוך הפוינטרים, ולא ערכי הצבעים של הפיקסלים עצמם. הפתרון הוא:

- למחוק את הכוכבית בשורה 16, כך שיהיה לנו מערך של פיקסלים ולא פוינטרים;
- למחוק את ה new בשורה 19;
- למחוק את ה delete בשורה 28 (לא צריך למחוק כי אחרי השינוי הכל נמצא על המחסנית).

שאלה 7 [20 נק']

כיתבו פונקציה בשם `find` המוצאת מילה בטקסט, גם אם סדר האותיות במילה התחלף.

למשל: נניח שהטקסט הוא `"tDno rryow eb phapy"`, אם המילה שצריך לחפש היא `"worry"`, אז הפלט יהיה `"rryow"` - זו אותה מילה רק בשינוי סדר האותיות (באנגלית זה נקרא אנגרמה - `anagram`).

ניתן להניח שבמחרוזת יש רק אותיות אנגליות גדולות וקטנות, ורווחים (ללא סימני פיסוק). יש לחפש רק מילים שלמות. יש להבחין בין אותיות גדולות לקטנות (למשל במחרוזת למעלה, נמצא `"Dont"` אבל לא נמצא `"dont"`).

מקרים מיוחדים:

- אם המילה שמחפשים אינה מופיעה בטקסט - יש לזרוק חריגה.
- אם המילה שמחפשים מופיעה פעמיים או יותר - יש להחזיר את ההופעה הראשונה.
- אם המילה שמחפשים היא ריקה, או שהיא לא מילה אחת (מכילה רווח) - יש לזרוק חריגה.

תוכנית ראשית לדוגמה:

```
#include "AnagramFinder.hpp"
#include <iostream>
#include <stdexcept>
using namespace std;

int main() {
    string text = "tDno rryow eb phapy";
    cout << anagram::find(text, "Dont") << endl;    // "tDno"
    cout << anagram::find(text, "worry") << endl;    // "rryow"
    cout << anagram::find(text, "be") << endl;        // "eb"
    cout << anagram::find(text, "happy") << endl;    // "phapy"
    try {
        cout << anagram::find(text, "dont") << endl;
        // should throw an exception - dont != Dont
    } catch (const exception& ex) {
        cout << "    caught exception: " << ex.what() << endl;
        // should print: "Word 'dont' is not found"
    }
    try {
        cout << anagram::find(text, "rry") << endl;
        // should throw an exception - not a whole word
    } catch (const exception& ex) {
        cout << "    caught exception: " << ex.what() << endl;
        // should print: "Word 'rry' is not found"
    }
}
```

חלק א [10 נק']. כתבו קובץ המבצע בדיקות-יחידה לפונקציה בעזרת מערכת **doctest** שהשתמשתם בה במטלות. יש לכתוב רק את הקובץ הכולל את הבדיקות עצמן (כמו הקובץ שהגשתם בחלק א של המטלה) - אין צורך לכתוב את התוכנית הראשית המריצה את הבדיקות (`TestRunner.cpp`). אין לחזור על בדיקות שנמצאות בתוכנית ההדגמה - תהיו מקוריים. הוסיפו הסברים והערות בקובץ כדי להסביר מה אתם בודקים. בשאלה זו מספיק לכתוב 15-20 בדיקות (CHECK) בסה"כ; כמובן יש לכסות את כל הדרישות של המטלה.

חלק ב [10 נק']. נתון מימוש לפונקציה find שבו הושמטו כמה חלקים ובמקומם נכתבו קוים תחתיים. השלימו את החסר בכל קו. אין צורך להעתיק את כל המימוש – אפשר לכתוב רק החלקים החסרים לפי המספר המופיע לידם. שימו לב: כל קו תחתי יכול לייצג מילה אחת או יותר.

המימוש פועל ע"פ האלגוריתם הבא:

- סדר את האותיות במילה שמחפשים;
- עבור כל מילה בטקסט:
 - 0 סדר את האותיות המילה;
 - 0 אם המילה המסודרת מהטקסט שווה למילה המסודרת שמחפשים – החזר את המילה המקורית מהטקסט.

הערה: במקום להשלים את החסר, מותר לכם גם לכתוב מימוש חדש, אולם במקרה זה עליכם להסביר בפירוט את האלגוריתם שהשתמשתם בו.

```
#include "AnagramFinder.hpp"
#include <sstream>
#include <algorithm>
#include <stdexcept>
using _____1;

namespace _____2 {
    _____3 find(_____4 text, _____5 wanted_word) {
        _____6;
        string text_word, sorted_text_word;
        istringstream text_stream(text);
        while (text_stream >> text_word) { // read the words in "text" one by one
            sorted_text_word = text_word;
            sort(_____7);
            if (_____8)
                return _____9;
        }
        _____10;
    }
}
```

פתרון אפשרי (ע"פ הרצאה 1 ומטלה 1, וגם ע"פ הרצאה 10):

א.

```
#include "doctest.h"
#include "AnagramFinder.hpp"
using namespace anagram;

#include <string>
using namespace std;

TEST_CASE("Word found") {
```



```

    string text = "abc def ghi";
    CHECK(find(text, "abc") == string("abc"));
    CHECK(find(text, "cba") == string("abc"));
    CHECK(find(text, "edf") == string("def"));
    CHECK(find(text, "igh") == string("ghi"));
}

TEST_CASE("Word not found") {
    string text = "abc DEF Ghi";
    CHECK_THROWS(find(text, "Abc"));
    CHECK_THROWS(find(text, "ABC"));
    CHECK_THROWS(find(text, "Def"));
    CHECK_THROWS(find(text, "def"));
    CHECK_THROWS(find(text, "GHI"));
    CHECK_THROWS(find(text, "ghi"));
}

TEST_CASE("Word found twice") {
    string text = "abc def bac ghi";
    CHECK(find(text, "cba") == string("abc"));
    CHECK(find(text, "abc") == string("abc"));
    CHECK(find(text, "bac") == string("abc"));
}

TEST_CASE("Illegal word") {
    string text = "abc DEF Ghi";
    CHECK_THROWS(find(text, ""));
    CHECK_THROWS(find(text, "a b"));
}

```

הערות:

- בשורה הרביעית בתוכנית הראשית היה באג – המילה שמחפשים היתה כתובה שם עם אות גדולה (Be) במקום עם אות קטנה (be). זאת בניגוד להסבר בעברית, האומר שיש להבחין בין אותיות גדולות לקטנות (ותודה רבה למתקן-השגיאות האוטומטי של "וורד", שמנסה לתקן לי באגים בדקדוק ומכניס לי באגים בקוד...). מי שהסביר בבירור, שבגלל התוכנית הראשית הוא חשב שצריך לא להבחין בין אותיות גדולות לקטנות, וכתב בדיקות בהתאם – לא יפסיד נקודות.
- כדי לבדוק אם פקודה זורקת חריגה יש להשתמש בפעולה CHECK_THROWS כמו ברוב הבדיקות שפורסמו. מי שלא השתמש בפקודה זו אלא הדפיס ל-cout הפסיד 3 נקודות – בבדיקות אוטומטיות אף-פעם לא מדפיסים.
- מי שלא השתמש ב TEST_CASE הפסיד 3 נקודות (כי מי שעשה חמש מטלות עם בדיקות לא היה שוכח..)

חלק ב.

1. namespace std;

2. anagram

3. string

4. string & const string&

5. string & const string&

6.

```
string sorted_wanted_word = wanted_word; sort(sorted_wanted_word.begin(), sorted_wanted_word.end());
```

7. sorted_text_word.begin(), sorted_text_word.end())

8. `sorted_text_word == sorted_wanted_word`

9. `text_word`

10. `throw invalid_argument("Word '"+wanted_word+"' is not found");`

[אפשר גם לזרוק חריגה אחרת – העיקר להשתמש ב `throw` ולכתוב את הודעת השגיאה המתאימה בפנים]

הערות:

- אין צורך לבדוק בפירוש שהמילה שמחפשים (`wanted_word`) לא ריקה או לא מכילה רווחים, כיוון שבלולאת `while` אנחנו בכל מקרה מוצאים רק מילים שלמות לא ריקות - כך עובד אופרטור הקלט. לכן אם נעביר מילה ריקה או עם רווחים הקוד בכל מקרה לא ימצא אותה ויזרוק חריגה.
- מי שבכל-זאת הוסיף פקודה לזריקת חריגה בשורה 6 – לא יפסיד נקודות.
- בסעיפים 6, 7 היה חשוב לבצע את הסידור בצורה נכונה בעזרת איטרטורים, כמו שלמדנו בהרצאה 10.

בתואר "התשובה המקורית ביותר" זוכה הפתרון של ----- שכתב:

```
1. aacdemnpst
2. aaagnmr
3. ginrst
4. ginrst
5. ginrst
```

שאלה 8 [20 נק']

כיתבו דמוי-מיכל בשם **range** המייצג טווח של מספרים שלמים. הטווח יכול לתמוך בסוגים שונים של מספרים שלמים, כגון: short, int, long (אין צורך לתמוך במספרים ממשיים). המספרים יכולים להיות מסודרים בסדר עולה או יורד.

לפניכם תוכנית ראשית לדוגמה:

```
#include <iostream>
#include <vector>
#include <string>
#include <ostream>
#include "range.hpp"

using namespace std;
using namespace itertools;

int main() {
    for (auto i: range<short>(5,9))
        cout << i << " ";    // 5 6 7 8
    for (auto i: range<short>(9,5))
        cout << i << " ";    // 9 8 7 6
    for (auto i: range<short>(9,9))
        cout << i << " ";    // Nothing is printed - empty range
    for (auto i: range<long>(1111111111,11111111114))
        cout << i << " ";    // 1111111111 11111111112 11111111113
}
```

עליכם לכתוב רק את הקובץ range.hpp.

פתרון אפשרי (ע"פ הרצאות 8-9 ומטלה 5):

```
#pragma once

namespace itertools {
    template<typename T>
    class range {
        T first, last, increment;
    public:
        range(T first, T last) : first(first), last(last) {
            increment = (last >= first? 1: -1);
        }
        class iterator {
            T current, increment;
        public:
            iterator(T current, T increment) :
                current(current), increment(increment) {}
            iterator& operator++() {
```

```

        current += increment;
        return *this;
    }
    iterator operator++(int) { // Not needed for the loop
        iterator previous = *this;
        current += increment;
        return previous;
    }
    bool operator==(const iterator &it) const { // Not needed for the loop
        return current == it.current;
    }
    bool operator!=(const iterator &it) const {
        return current != it.current;
    }
    T operator*() {
        return current;
    }
};

iterator begin() const{
    return iterator{first,increment};
}

iterator end() const {
    return iterator{last,increment};
}

};
}

```

מפתח ניקוד:

- המחלקה range עצמה – 10: תבנית, increment, בנאי, begin, end – כל אחד 2 נקודות.
- המחלקה הפנימית 10 – iterator: שדות, בנאי, ++, !=, * - כל אחד 2 נקודות.

הערות:

- היו סטודנטים שחשבו, שבגלל שנאמר בשאלה לכתוב רק את הקובץ range.hpp, שצריך לכתוב רק את הכותרות של השיטות בלי המימושים שלהן. זו טעות - כמו שלמדנו בהרצאות, כשכותבים מחלקה שהיא תבנית (template), כותבים את כולה בקובץ ה-hpp, כיוון שהקומפיילר צריך להכיר את כל המימוש בזמן קומפילציה.
- עם זאת, כיוון שגם לדעת מה הכותרות הנכונות זה לא פשוט, ומראה על הבנה של החומר, יינתן ניקוד חלקי גם למי שכתב כותרות בלבד – בתנאי כמובן שהכותרות נכונות ומלאות.
- מימוש המשתמש באופרטור - - לא יעבוד כאן, כי הלולאה בתוכנית הראשית לא קוראת בכלל לאופרטור --. לולאת for כמו בתוכנית הראשית קוראת תמיד לאופרטור ++ בלבד. לכן צריך לממש את האופרטור ++ באופן המתייחס לכיוון התנועה – קדימה או אחורה.
- אין צורך לממש אופרטור פלט ל range – לא משתמשים בו בשום מקום.
- אין לשים משתנה מסוג vector במחלקה range – זה מאד לא יעיל.

