פקולטה: מדעי הטבע
מחלקה: מדעי המחשב
שם הקורס: תכנות מערכות ב
קוד הקורס: 7023010-2 כל הקבוצות
מועד __**א**___ סמסטר: ___**ב**___ שנה: ___ה'תשפ"א
תאריך הבחינה: ו' בתמוז ה'תשפ"א, 16/06/2021
משך הבחינה: שעתיים וחצי – 150 דקות
המרצים: אראל סגל-הלוי, ערן קאופמן, חרות סטרמן.
המתרגלים: יבגני נייטרמן, אריה יטיב, אור אביטל.

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 [8 נק'] - אופרטורים

כתבו מחלקה בשם Box המייצגת תיבה שיש לה אורך (length), רוחב (breadth) וגובה (height). הממדים הם מספרים שלמים. כתבו אופרטור קטן ">" המשווה בין שתי תיבות לפי האורך; אם האורך זהה, משווה לפי הרוחב; אם גם הרוחב זהה, משווה לפי הגובה. לפניכם תוכנית ראשית לדוגמה; יש לממש את כל השיטות הדרושות כדי שהתוכנית תעבוד לפי ההוראות. אין צורך לממש שיטות שאינן נזכרות בתוכנית הראשית.

```
#include <iostream>

using namespace std;

int main() {
      Box b1(1, 2, 4); // A box with length 1, breadth 2, height 4
      Box b2(2, 1, 4); // A box with length 2, breadth 1, height 4
      Box b3(2, 2, 3); // A box with length 2, breadth 2, height 3
      cout << boolalpha << (b1 < b2) << endl;
          // prints "true" since the length of b1 is smaller.
      cout << boolalpha << (b2 < b3) << endl;
          // prints "true" since
          // the length is equal, but the breadth of b2 is smaller.
      cout << boolalpha << (b3 < b1) << endl;
          // prints "false" since the length of b3 is larger.
      cout << boolalpha << (b3 < b3) << endl;
          // prints "false"
}
```

שאלה 2 [9 נק'] – פאנדמיק

הוסיפו לפתרון מטלה 4 שלכם, שחקן מסוג חדש בשם "ילד" – Child. התכונה המיוחדת של השחקן היא, שהוא יכול להחזיק ביד בכל עת רק לכל היותר שישה קלפים. אם יש לו כבר שישה קלפים והוא מנסה לקחת עוד קלף, הוא לא יקבל קלף חדש אלא מצבו יישאר ללא שינוי (לא תיזרק חריגה). הנה תוכנית ראשית לדוגמה:

```
#include "Board.hpp"
#include "City.hpp"
#include "Child.hpp"
using namespace pandemic;

void test(Player& player) {
      player
      .take_card(City::Atlanta)
      .take_card(City::Santiago)
      .take_card(City::BuenosAires)
      .take_card(City::Tehran)
      .take_card(City::Baghdad)
      .take_card(City::Algiers)
      .take_card(City::Paris); // this card is not taken (no exception).
      // player.fly_direct(City::Paris);      // throws an exception –
                        // the child does not have the Paris card.
      player.fly_charter(City::Taipei);  // uses the Atlanta card.
      player.take_card(City::Paris); // this card is taken
      player.fly_direct(City::Paris); // no exception
}

int main() {
      Board board;
      Child child {board, City::Atlanta};
      test(child);
}
```

א [3 נק']. הסבירו בפירוט מה בדיוק צריך לשנות בקבצים **הקיימים** של המטלה שלכם. עבור כל שורה שצריך לשנות, כתבו באיזה קובץ היא נמצאת, מה השורה לפני השינוי, מה השורה לאחר השינוי, ומדוע השינוי דרוש.

ב [3 נק']. כתבו בפירוט את הקובץ Child.hpp עם הכותרות בלבד.

ג [3 נק']. כתבו בפירוט את הקובץ Child.cpp עם כל המימושים.

תזכורת: תקציר ההוראות למטלה 4

*[**אין צורך** לכתוב פתרון מלא למטלה במסגרת הבחינה. התזכורת נועדה רק למי ששכח את פרטי המטלה].*

אחד המשחקים שנוצרו כדי להתאמן בהתמודדות עם מגפות הוא פאנדמיק. זהו משחק לוח ל-2 עד 4 שחקנים, המשחקים במשותף כדי לרפא מחלות ולגלות תרופות. במטלה זו נממש חלק מחוקי המשחק, עבור שחקן אחד בלבד.

הלוח

לוח המשחק הוא מפה של העולם, ובה 48 ערים. חלק מהערים מחוברות בקו המאפשר לנסוע ביניהן. הערים מחולקות לארבעה צבעים - כחול, צהוב, שחור ואדום - 12 ערים מכל צבע. במטלה זו, הלוח מיוצג ע"י המחלקה Board.

הקלפים

ישנם 48 קלפים - קלף אחד לכל עיר. שמות הערים בקוד יהיו בדיוק כמו על המפה, בלי הרווחים. לדוגמה, השם של העיר ניו יורק הוא NewYork. יש להקפיד על האיות על-מנת למנוע שגיאות קומפילציה.

המחלות

ישנן מחלות מארבעה צבעים - כחול, צהוב, שחור ואדום. בכל עיר עשויות להיות "קוביות מחלה" בצבע המתאים לאותה עיר. במחלקה "לוח" יש לממש את השיטות הבאות:

- *אופרטור סוגריים מרובעים [] - מקבל כפרמטר מזהה-עיר, ומאפשר לקרוא ולעדכן את רמת המחלה (=מספר קוביות המחלה) באותה עיר. לדוגמה: board[City::HongKong] = 2 מציבה שתי קוביות-מחלה אדומות בעיר הונג קונג.*
  - *אופרטור פלט - מציג את מצב הלוח בפורמט כלשהו לבחירתכם. מצב הלוח כולל את רמת המחלה בכל אחת מהערים, את התרופות שהתגלו עד כה, ואת תחנות-המחקר שנבנו עד כה.*
- *is_clean — שיטה בוליאנית ללא פרמטרים, המחזירה "אמת" אם ורק אם כל הלוח נקי - אין שום קוביות מחלה.*
- *remove_cures - - שיטה ללא פרמטרים, המסירה מהלוח את כל התרופות שהתגלו עד כה. שיטה זו נועדה לצורך כתיבת בדיקות; היא אף פעם לא זורקת חריגה, ואין צורך לבדוק אותה באופן מיוחד.*

השחקן

השחקן מתחיל את המשחק באחת הערים, ומקבל מספר קלפים. בכל תור, הוא יכול לבצע אחת מהפעולות הבאות:

1. *נסיעה - drive - מעבר מהעיר הנוכחית לעיר סמוכה (בהתאם למפת הקשרים).*
2. *טיסה ישירה - fly_direct - מעבר מהעיר הנוכחית לעיר של קלף כלשהו שנמצא בידו. כדי לבצע פעולה זו יש להשליך את הקלף המתאים לעיר שטסים אליה.*
3. *טיסת זכיון - fly_charter - מעבר מהעיר הנוכחית לעיר כלשהי. כדי לבצע פעולה זו יש להשליך את הקלף המתאים לעיר שנמצאים בה.*
4. *טיסת שאטל - fly_shuttle - אם בעיר הנוכחית ישנה תחנת-מחקר, אפשר לטוס לכל עיר אחרת שיש בה תחנת-מחקר.*
5. *בנייה - build - בניית תחנת-מחקר בעיר שנמצאים בה. כדי לבצע פעולה זו יש להשליך את הקלף המתאים לעיר שנמצאים בה.*
   - *בכל עיר יכולה להיות לכל היותר תחנת-מחקר אחת. אם כבר יש תחנת מחקר בעיר הנוכחית, ומבצעים שוב פעולת "בנייה", אין צורך לזרוק חריגה, והקלף נשאר בידי השחקן.*

6. גילוי תרופה - discover_cure - גילוי תרופה למחלה בצבע מסויים. כדי לבצע פעולה זו, יש להימצא בעיר שיש בה תחנת-מחקר, ולהשליך 5 קלפים בצבע של המחלה. צבע העיר שנמצאים בה לא משנה.
   - לכל מחלה יש תרופה אחת. אם כבר התגלתה תרופה למחלה, ומבצעים שוב פעולת "גילוי תרופה" לאותה מחלה, אין צורך לזרוק חריגה, והקלפים נשארים בידי השחקן.
7. טיפול במחלה - treat - הורדת קוביית-מחלה אחת מהעיר שנמצאים בה (הפחתת רמת המחלה ב-1).
   - אם כבר התגלתה תרופה למחלה בצבע של העיר, אז פעולת "ריפוי מחלה" מורידה את כל קוביות המחלה מהעיר שנמצאים בה (הפחתת רמת המחלה ל-0).
   - אם אין בכלל זיהום בעיר (רמת המחלה היא 0), אז הפעולה תזרוק חריגה.

כל פעולה צריכה לעדכן את מצב הלוח ומיקום השחקן בהתאם. אם הפעולה לא חוקית, יש לזרוק חריגה מתאימה.

בנוסף, יש לממש את השיטות הבאות:

- role –- פונקציה המחזירה את התפקיד של השחקן (ראו רשימת התפקידים למטה), לצורך תצוגה.
- take_card - - לקיחת קלף-עיר כלשהו. פעולה זו מדמה את התהליך שבו השחקן מקבל קלפים מהקופה בתחילת המשחק או במהלכו.
  - יש רק קלף אחד של כל עיר, ולכן אם מבצעים פעולת take_card על קלף שכבר נמצא בידי השחקן, לא יהיה כל שינוי במצב השחקן.

תפקידים

ישנם תפקידים שונים של שחקנים, שיש להם כישורים מיוחדים (הכישורים דומים אבל לא זהים למשחק המקורי):

1. קצין מבצעים - OperationsExpert:יכול לבצע פעולת "בנייה" בכל עיר שהוא נמצא בה, בלי להשליך קלף-עיר מתאים.
2. קצין תעבורה - Dispatcher: כשהוא נמצא בתחנת-מחקר, הוא יכול לבצע פעולת "טיסה ישירה" לכל עיר שהוא רוצה, ללא השלכת קלף-עיר.
3. מדענית - Scientist: יכולה לבצע פעולת "גילוי תרופה" בעזרת n קלפים בלבד (במקום 5), כאשר הפרמטר n מועבר בבנאי (במשחק המקורי n=4).
4. חוקרת - Researcher: יכולה לבצע פעולת "גילוי תרופה" בכל עיר - לא חייבת להיות בתחנת מחקר.
5. פראמדיק - Medic: כשהוא מבצע פעולת "טיפול במחלה", הוא מוריד את כל קוביות-המחלה מהעיר שהוא נמצא בה, ולא רק אחת.
   - אם כבר התגלתה תרופה למחלה, הוא אוטומטית מוריד את כל קוביות-המחלה מכל עיר שהוא נמצא בה, גם בלי לבצע פעולת "טיפול במחלה".
6. וירולוגית - Virologist: יכולה לבצע פעולת "טיפול במחלה", לא רק בעיר שהיא נמצאת בה, אלא בכל עיר בעולם - ע"י השלכת קלף של אותה העיר.
7. פורסת גנים - GeneSplicer: יכולה לבצע פעולת "גילוי תרופה" בעזרת 5 קלפים כלשהם - לא דווקא מהצבע של המחלה.
8. רופא שטח - FieldDoctor: יכול לבצע פעולת "טיפול במחלה" לא רק בעיר שהוא נמצא בה אלא בכל עיר סמוכה לעיר שהוא נמצא בה (ע"פ מפת הקשרים), בלי להשליך קלף עיר.
9. **התפקיד החדש עבור הבחינה: ילד - Child: יכול להחזיק רק שישה קלפים בכל עת. אם הוא מנסה לקחת קלף נוסף, הוא לא מצליח, ומצב הקלפים שלו נשאר כפי שהיה קודם (לא נזרקת חריגה).**

שאלה 3 [9 נק'] – בדיקות-יחידה – עץ בינארי

נתבקשתם לשנות את העץ הבינארי שכתבתם במטלה 5, כך שלולאת range-based for loop תעבוד ע"פ הסדר הבא: קודם הילד **הימני**, אחר-כך ההורה, ואחר-כך הילד **השמאלי**. לדוגמה:

```
int main() {
      BinaryTree<int> tree_of_ints;
      tree_of_ints.add_root(1)
      .add_left(1, 2)
      .add_left(2, 4)
      .add_right(2, 5)
      .add_right(1, 3);

      /* The tree now looks like this:
            1
            |--------|
            2        3
            |---|
            4   5
      */

      for (int element : tree_of_ints) {
            cout << element << " ";
      } // prints: 3 1 5 2 4
}
```

כתבו קובץ המבצע **בדיקות-יחידה מקיפות** עבור השינוי בעזרת מערכת doctest. חלקו את הבדיקות למקרי-בדיקה הגיוניים, והסבירו מה בודק כל אחד מהם.

- **אין צורך** לבדוק פונקציות שלא השתנו (כגון add_root).
- **אין צורך** לכתוב את המימוש עצמו אלא רק את הבדיקות.

שאלה 4 [6 נק'] - לינוקס

כתבו פקודות בלינוקס המבצעות את המשימות הבאות (כל משימה בפקודה אחת בלבד):

א [2 נק']. כתיבת רשימת הקבצים בתיקיה הנוכחית לתוך קובץ בשם `files.txt`.

ב [2 נק']. ספירת מספר הקבצים בתיקיה הנוכחית והדפסת המספר על המסך.

ג [2 נק']. ספירת מספר הקבצים בתיקיה הנוכחית וכתיבת המספר לתוך קובץ בשם `filecount.txt`.

לדוגמה: אם בתיקיה הנוכחית נמצאים הקבצים הבאים:

```
Board.hpp  Board.cpp   Test.cpp
```

אז בסעיף א יש לכתוב לקובץ את הרשימה "`Board.hpp  Board.cpp   Test.cpp`", בסעיף ב יש לכתוב למסך את המספר 3, ובסעיף ג יש לכתוב לקובץ את המספר 3.

- אם אתם לא זוכרים שם של פקודה מסויימת בלינוקס, נסו לכתוב לפי מיטב זכרונכם, והסבירו בעברית למה התכוונתם.

שאלה 5 [9 נק'] – אלגוריתמים מהספרייה התקנית

השאלה מתייחסת לפונקציה מהספרייה התקנית, המוגדרת באופן הבא (מתוך התיעוד של השפה):

```
template< class ForwardIt >
constexpr std::pair<ForwardIt,ForwardIt>
    minmax_element( ForwardIt first, ForwardIt last );
```

```
01   #include <iostream>
02   #include <algorithm>
03   #include <vector>
04   using namespace std;
05
06   class Person {
07       private:
08           string _name;
09           int _age;
10       public:
11           Person(const string& name, int age): _name(name), _age(age) {}
12           int age() { return _age; }
13   };
14
15   vector<Person> v {{"Avi",11},{"Beni",22},{"Chana",44},{"Dani",33}};
16   int main() {
17       auto [min,max] = minmax_element(v.begin(), v.end(),
```

```
18            [](Person a, Person b){return a.age()<b.age();} );
19        cout << "The youngest person is: " << min << endl;
20        cout << "The oldest person is: " << max << endl;
21    }
```

התוכנית אמורה להדפיס את השם והגיל של האיש הצעיר ביותר והזקן ביותר בוקטור. למשל בתוכנית הנתונה מודפס:

```
The youngest person is: Avi (11)
The oldest person is: Chana (44)
```

(כמובן הפלט אמור להשתנות בהתאם לוקטור). אולם כשמריצים את התוכנית, מקבלים הודעת שגיאה ארוכה שמתחילה כך:

```
Person_minimax.cpp:19:40: error: invalid operands to binary expression ('ba-
sic_ostream<char, std::char_traits<char> >' and 'std::tuple_element<0, std::
pair<__gnu_cxx::__normal_iterator<Person *, std::vector<Person, std::alloca-
tor<Person> > >, __gnu_cxx::__normal_iterator<Person *, std::vector<Person,
std::allocator<Person> > > > >::type' (aka '__gnu_cxx::__normal_iterator<Per
son *, std::vector<Person, std::allocator<Person> > >'))
    cout << "The youngest person is: " << min << endl;
    ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ ^  ~~~
/usr/bin/../lib/gcc/x86_64-linux-gnu/9/../../../../include/c++/9/
ostream:245:7: note: candidate function not viable: no known conversion from
'std::tuple_element<0, std::pair<__gnu_cxx::__normal_iterator<Person *, std:
:vector<Person, std::allocator<Person> > >, __gnu_cxx::__normal_iterator<Per
son *, std::vector<Person, std::allocator<Person> > > > >::type' (aka '__gnu
_cxx::__normal_iterator<Person *, std::vector<Person, std::allocator<Person>
> >') to 'const void *' for 1st argument; take the address of the argument w
ith &
      operator<<(const void* __p)
      ^
...
```

א [3 נק']. הסבירו בפירוט את הודעת השגיאה. מה משמעות ההודעה ומה גורם לה?

כדי לתקן את השגיאה בצורה נכונה, כך שהתוכנית תמלא את תפקידה, צריך לעשות שינוי קטן בתוכנית הראשית (להוסיף עד 6 תווים בלבד), **וגם** להוסיף עוד קוד מחוץ לתוכנית ראשית.

ב [3 נק']. מה השינוי שצריך לעשות בתוכנית הראשית (להוסיף לכל היותר 6 תווים)? כתבו את הקוד לאחר השינוי.

ג [3 נק']. איזה קוד צריך להוסיף מחוץ לתוכנית הראשית?

שאלה 6 [9 נק'] – מצביעים חכמים איטרטורים וטמפלייטינג

לפניכם מחלקה בשם stack (מחסנית), אשר ממומשת ע"י רשימה מקושרת של אלמנטים. למחלקה יש מתודת push להוספת int לראש התור, מתודת pop להוציא מראש התור.

```
class Stack {

      struct Node{
            int value;
            Node* next;
      };

      Node* head = nullptr;

public:
      void push(int v){
            Node* newNode = new Node;
            newNode->value=v;
            newNode->next=head;
            head = newNode;
      }

      int pop(){
            Node* nodeToDel = head;
            int val = head->value;
            head = head->next;
            delete nodeToDel;
            return val;
      }

      ~Stack() {
            while (head!=nullptr) { pop(); }
      }
{;
```

שנו את המחלקה לפי הכללים הבאים:

- א [2 נק']. המחלקה תהיה מטמופלטת (template) ותוכל להכיל נתונים מכל סוג שהוא.
- ב [2 נק']. המחלקה תשתמש במצביעים חכמים במקום במצביעים רגילים (אין להשתמש בסימון למצביע * או בקריאה ל new או delete).
- ג [2 נק']. המחלקה תכלול מתודה בשם del, אשר תקבל אלמנט מסוים, ואם תמצא אותו ברשימה - תמחק אותו ותשחרר את כל הרשימה החל מהאלמנט הזה ועד הסוף (אם האלמנט לא נמצא, לא יקרה כלום – אין צורך לזרוק חריגה).
- ד [3 נק']. המחלקה תאפשר מעבר על כל האיברים ב range-based for loop.

תוכנית ראשית לדוגמא:

```
int main(){

    Stack<int> s;
    s.push(1);
```

```
    s.push(2);
    s.push(3);
    s.push(4);

    for(auto& i : s) cout << i << " ";
    cout << endl;   // prints 4 3 2 1

    s.del(2);

    for(auto& i : s) cout << i << " ";
    cout << endl;    // prints 4 3
{
```

שאלה 7 [2 נק']

יינתן מענק של 2 נקודות על כתיבה מסודרת לפי הפירוט הבא:

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