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.

פתרון הבחינה
שימו לב: מפתח הניקוד לכל שאלה הוא בגדר טיוטה בלבד – הוא עשוי להשתנות בהמשך.
מפתח ניקוד מעודכן יפורסם לאחר בדיקת הבחינות.

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"
}
```

פתרון אפשרי (כמו במטלת האקראנק הראשונה):

```
class Box {
      private:
            int l, b, h;
      public:
            Box(int l, int b, int h): l(l), b(b), h(h) {}

      bool operator<(const Box& other) const {
            if(l<other.l){
                  return true;
            } else if(l==other.l && this->b<other.b){
                  return true;
            } else if(l==other.l && this->b==other.b && this->h<other.h){
                  return true;
            } else {
                  return 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.

פתרון: צריך לשנות בקובץ Player.hpp, במחלקה Player, את השורה:

- `Player& take_card(City city);`
- לשורה:
- `virtual Player& take_card(City city);`
- זאת על-מנת שהשיטה תהיה וירטואלית ויהיה אפשר להחליף אותה במחלקה היורשת.

הערה: יש סטודנטים, שכבר בפתרון שהגישו, השיטה הזאת היתה וירטואלית. סטודנטים כאלו יקבלו את מלוא הנקודות אם יציינו שלא צריך לשנות שום דבר.

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

פתרון אפשרי:

```
#include "Board.hpp"
#include "City.hpp"
#include "Player.hpp"
#include <string>

namespace pandemic {
      class Child: public Player {
            public:
                  Child(Board& board, City city);
                  Player& take_card(City city);
                  std::string role();
      };
}
```

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

פתרון אפשרי:

```
#include "Child.hpp"

const int MAX_CARDS = 6;
namespace pandemic {
      Child::Child(Board& board, City city): Player(board,city) {}
      Player& Child::take_card(City city) {
            if (this→cards.size() <  MAX_CARDS) {
                  Player::take_card(city);
            }
            return *this;
      }
      Child::role() { return “Child”; }
}
```

הערה לסעיפים ב-ג: *יש שמימשו את השיטה role בצורה אחרת – ע"י משתנה פרטי שמועבר לבנאי של* Player. בפתרון זה אין צורך לכתוב את השיטה role בקובץ הכותרת, אבל יש צורך להעביר פרמטר "Child" בקריאה לבנאי בקובץ המימוש.

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.

פתרון אפשרי: הנה דוגמה לבדיקה של עץ מספרים:

```
#include "doctest.h"
#include "BinaryTree.hpp"

TEST_CASE("Tree of ints") {
      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);
```

```
    int expected[5] {3,1,5,2,4};
    int index = 0;
    for (auto element : tree_of_ints) {
        CHECK_EQ(element, expected[index++]);
    }
}
```

באותו אופן יש לבדוק גם עצים של סוגים נוספים (למשל מחרוזת, עצם), וכן עצים מורכבים יותר.
כמובן ישנן דרכים רבות לבצע בדיקה זו, וכל פתרון ייבחן לגופו.

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.

פתרון: הדבר העיקרי שהיה צריך לזכור זה את האופרטורים להכוונת קלט ופלט ו"צינור":

א.

```
ls > files.txt
```

ב.

```
ls | wc -l
```

או לחילופין

```
ls | wc -w
```

ג.

```
ls | wc -l > filecount.txt
```

או לחילופין

```
ls | wc -w > filecount.txt
```

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 );
```

תיקון: לפונקציה יש הגדרה נוספת:

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

```
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?

פתרון: הודעת השגיאה אומרת "**invalid operands to binary expression**". בעברית: "אופרנדים לא חוקיים לביטוי בינארי". המשמעות היא, שיש לנו בקוד ביטוי עם אופרטור בינארי, אבל האופרנדים לא מתאימים לאופרטור. במקרה זה, האופרטור הוא אופרטור הפלט (מסומן בחץ). האופרנדים שלו הם: ערוץ פלט (basic_ostream עם ארגומנט הוא למעשה הסוג של ostream), ואיטרטור.

המשתנים min, max הם למעשה איטרטורים שמצביעים לתוך הוקטור (כדי לדעת את זה צריך להבין איך עובדים האלגוריתמים בספריה התקנית – כל האלגוריתמים עובדים עם איטרטורים). אנחנו מנסים להדפיס אותם, אבל אין אופרטור פלט לאיטרטורים, ולכן נוצרת השגיאה.

הערה: כיוון שהיתה טעות בהגדרת הפונקציה בגוף השאלה, יינתנו נקודות גם לסטודנטים שפתרו את השאלה ע"פ ההגדרה השגויה (עם שני ארגומנטים), בתנאי שהפתרון עצמו נכון.

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.

פתרון: בגוף התוכנית הראשית, צריך להדפיס לא את האיטרטור, אלא את הדבר שהאיטרטור מצביע עליו, ולשם כך יש להשתמש באופרטור כוכבית:

```
22        cout << "The youngest person is: " << (*min) << endl;
23        cout << "The oldest person is: " << (*max) << endl;
```

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

פתרון: כדי שהקומפיילר יידע להדפיס אדם, צריך להוסיף אופרטור פלט:

```
ostream& operator<< (ostream& out, const Person& p) {
    return (out << p._name << " (" << p._age << ")");
}
```

בנוסף, כיוון שהשדות פרטיים, צריך להוסיף שורה בגוף המחלקה כדי שהאופרטור יוכל לגשת אליהם:

```
friend ostream& operator<< (ostream& out, const Person& p);
```

לחלופין, ניתן להוסיף שיטה ציבורית name() ולהשתמש בה.

הערה: למרות שכתבנו בפירוש "איזה **קוד**", היו סטודנטים שחשבו שלא צריך לכתוב קוד. סטודנטים אלו יקבלו 2 נקודות מתוך 3 – אחת על ציון האופרטור ואחת על ציון ה-friend.

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
{
```

פתרון אפשרי: השורות שהשתנו מסומנות במספר הסעיף המתאים.

אנו משתמשים במצביע "משותף" (shared_ptr) כיוון שצריך שני מצביעים בו-זמנית לאותו זיכרון.

שימו לב שבפתרון החדש **לא צריך** מפרק.

```
using namespace std;
template<typename T>    // א
class Stack {

      struct Node{
            T value;     // א
            shared_ptr<Node> next;   // ב
      };

      shared_ptr<Node> head;        // ב

public:
      void push(T value){
            auto newNode = make_shared<Node>();  // ב
            newNode->value = value;
            newNode->next = head;
            head = newNode;
      }

      T pop() {
            head = head->next;
      }

      shared_ptr<Node> del(shared_ptr<Node> curr, T value){ // ג
            if (curr == nullptr) return nullptr;
            if (curr->value == value) return nullptr;
            curr->next = del(curr->next, value);
            return curr;
      }

      void del(T value){                // ג
            head=del(head, value);
      }
```

```
    struct iterator{  // ד

        shared_ptr<Node> curr;
        iterator(shared_ptr<Node> a):curr(a){};

        void operator++(){
            curr = curr->next;
        }
        bool operator!=(iterator& o){
            return curr != o.curr;
        }
        int& operator*(){
            return curr->value;
        }
    };

    iterator begin(){   // ד
        return head;
    }

    iterator end(){     // ד
        return nullptr;
    }

};
```

הערה: יתקבלו גם פתרונות שמימשו כל סעיף בנפרד.