פקולטה: מדעי הטבע
מחלקה: מדעי המחשב
שם הקורס: מבנה זיכרון ושפת C++
קוד הקורס: 2-7027810 כל הקבוצות - קבוצות 1,3,4,5
מועד __**ב**___ סמסטר: ___ב___ שנה: ___ה'תשעט
תאריך: י"א בתמוז ה'תשע"ט, 14/7/2019
משך הבחינה: 3 שעות
שם המרצים: אראל סגל-הלוי, גיל בן-ארצי

יש לענות על כל השאלות במחברת הבחינה.

אין צורך להעתיק את השאלון למחברת - השאלון יתפרסם במודל לאחר הבחינה.

יש לענות תשובות מלאות אך ממוקדות. לא יינתנו נקודות על תשובות עם טקסט מיותר שאינו קשור לנושא.

אסור להשתמש בכל חומר עזר.

בהצלחה!!!

שאלה 1 [10 נק']

```
01  #include <iostream>
02  using namespace std;
03
04  class Base {
05   public:
06    Base(){}
07    void Print( ){
08        cout <<" I am base "<<endl;
09    }
10    Base& operator=(const Base & b){
11       return this;
12    }
13  };
14
15  class Derived : public Base {
16   public:
17    Derived():Base(){
18    }
19
20    void Print( ){
21        cout <<" I am derived "<<endl;
22    }
23    Derived& operator=(const Derived &){
24       return this;
25    }
26  };
```

עבור סעיפים א-ב, התייחסו לתוכנית הראשית הבאה:

```
int main() {
    Derived d;
    Base b;
    b=d;
    b.Print();
    return 0;
}
```

א [3 נק']. בקוד יש שתי שגיאות קומפילציה. מה הן, ואיך אפשר לתקנן? אין למחוק קוד ואין לשנות את main.

ב [2 נק']. מה יהיה הפלט לאחר התיקון?

עבור סעיפים ג-ד, הניחו שהשגיאות מהסעיפים הקודמים תוקנו, והתייחסו לתוכנית הראשית הבאה:

```
int main() {
    Derived d;
    Base b;
    b=d;
    b.Print();
    d=b;
    d.Print();
    return 0;
}
```

ג [3 נק']. בקוד יש שגיאת קומפילציה אחת. מה היא, ואיך אפשר לתקנה? אין לשנות את main.

ד [2 נק'] מה יהיה הפלט לאחר התיקון?

שאלה 2 [9 נק']

```
01   #include <iostream>
02   #include <string>
03   using namespace std;
04
05   class Player {
06       public:
07           void play() {
08               cout << "Error: Player.play is not defined" << endl;
09           }
10   };
11
12   class SmartPlayer: public Player {
13       public:
14           void play() {
15               cout << "Smart Move" << endl;
16           }
17   };
18
19   class SimplePlayer: public Player {
20       public:
21           void play() {
22               cout << "Simple Move" << endl;
23           }
24   };
25
26   void game(Player* player1, Player* player2, int turns) {
27       for (int i=0; i<turns; ++i) {
28           player1->play();
29           player2->play();
30       }
31   }
32
33   int main() { // a demo program
34       game(new SmartPlayer(), new SimplePlayer(), 5);
35   }
```

כשמריצים את התוכנית, רואים 10 פעמים את המחרוזת:

```
Error: Player.play is not defined
```

א [3 נק']. מה סוג השגיאה (קומפילציה / קישור / זמן-ריצה / לוגית)? מה בדיוק גורם לשגיאה?

ב [3 נק']. איך אפשר לתקן את הבאג ע"י שינוי/הוספה של שורה אחת בלבד? ציינו את מספר השורה שיש לשנות/להוסיף והסבירו מדוע השינוי פותר את הבעיה. *אין לשנות את התוכנית הראשית.*

ג [3 נק']. מה יהיה פלט התוכנית לאחר התיקון?

שאלה 3 [8 נק']

```
1. #include <iostream>
2. using namespace std;
3.
4. void change(int*& ip, int j) {
5.    ip = (int *)(++j);
6. }
7.
8. int main() {
9.    int* ip = 0,j=0;
10.   change(ip,j);
11.   cout << "ip = " << ip << " j="<< (j++) <<endl;
12.   cout << "ip = " << (++ip) << " j="<< j <<endl;
13.   return 0;
14. }
```

א [4 נק']. האם יש שגיאת קומפילציה או אזהרה בתוכנית? אם כן ציינו היכן ותקנו אותה.

ב [4 נק']. מה פלט התוכנית?

שאלה 4 [9 נק']

נתונה התוכנית:

```
01   #include <iostream>
02   #include <algorithm>
03   #include <vector>
04   using namespace std;
05
06   struct Person {
07       string name;
08       int age;
09   };
10
11   int main() {
12       vector<Person> unsorted { {"bbb",11}, {"aaa",33}, {"ddd",22}, {"ccc",44}  };
13       vector<Person> sorted_by_age = unsorted;
14       sort(sorted_by_age.begin(), sorted_by_age.end());
15       for (Person p: sorted_by_age)
16            cout << p.age << " ";
17   }
```

כשמקמפלים את התוכנית מתקבל פלט ארוך מאד, המתחיל ב:

```
/usr/bin/../lib/gcc/x86_64-linux-gnu/7.3.0/../../../../include/c++/7.3.0/bits/predefined_ops.h:43:23: error:
invalid
      operands to binary expression ('Person' and 'Person')
      { return *__it1 < *__it2; }
        ~~~~~~ ^ ~~~~~~
```

ומסתיים ב:

```
question.cpp:14:8: note: in instantiation of function template specialization
       'std::sort<__gnu_cxx::__normal_iterator<Person *, std::vector<Person,
std::allocator<Person> > > >' requested here
        sort(sorted_by_age.begin(), sorted_by_age.end());
        ^
```

א [3 נק']. מה סוג השגיאה? מדוע הודעת השגיאה (**error**) מפנה לקובץ שלא אנחנו כתבנו? ומדוע אחרי הודעת-השגיאה יש **note** שמפנה לקוד שאנחנו כתבנו?

ב [3 נק']. הוסיפו קטע קוד *לפני התוכנית הראשית*, כך שהתוכנית תתקמפל ותרוץ והוקטור `sorted_by_age` אכן יכיל את רשימת האנשים מסודרים לפי גיל (מהצעיר למבוגר). *בסעיף זה אין לשנות את התוכנית הראשית.*

ג [3 נק']. הוסיפו קוד *בשורה 14 בלבד*, כך שהתוכנית תתקמפל ותרוץ והוקטור `sorted_by_age` אכן יכיל את רשימת האנשים מסודרים לפי גיל (מהצעיר למבוגר). *בסעיף זה אין לשנות שורות אחרות.*

שאלה 5 [22 נק']

סטודנט כתב מחלקה בשם PhysicalNumber המייצגת מספר פיסיקלי עם יחידות, היכול לתמוך בחיבור וחיסור **וגם כפל וחילוק**. כיתבו בדיקות-יחידה מקיפות עבור המחלקה. יש לבדוק את הדרישות הבאות:

- תמיכה בארבעה אופרטורים חשבוניים + - * / , וכן אופרטור הפלט.
- תמיכה בשני סוגים של יחידות אורך – מטר וקילומטר, ושני סוגים של יחידות זמן – שעה ודקה.
- כפל וחילוק אפשריים **תמיד** – בין כל סוגי היחידות. היחידות של התוצאה נקבעות לפי היחידות של המספרים המוכפלים/מחולקים. למשל, אם מחלקים 100 קילומטר ב-2 שעות, התוצאה היא "50 קילומטר / שעה". אם מכפילים "50 קילומטר / שעה" ב-30 דקות, התוצאה היא "25 קילומטר", וכו'.
- חיבור וחיסור – אפשרי רק בין מספרים עם יחידות תואמות, כגון: אורך+אורך, זמן+זמן, אורך/זמן + אורך/זמן, וכו'.
- חיבור וחיסור של מספרים עם יחידות לא-תואמות, למשל אורך + אורך/זמן, יגרום לחריגה.

א [11 נק']. **תיכנון הבדיקות**: כיתבו (בעברית) רשימה מפורטת של מקרים שאתם מתכוונים לבדוק.

ב [11 נק']. **מימוש הבדיקות**: מצורף קובץ עם בדיקות לדוגמה. עליכם להרחיב אותו ולהוסיף לו את הבדיקות בהתאם לתיכנון שכתבתם בסעיף א.

הפונקציות CHECK_OK, CHECK_EQUAL , CHECK_THROWS כבר מוגדרות בקובץ badkan.hpp, כמו במטלות. אין צורך לממש אותן. מותר להוסיף משתנים לפי הצורך.

```
#include <iostream>
#include "badkan.hpp"
#include "PhysicalNumber.hpp"
using namespace std;
using ariel::PhysicalNumber, ariel::Unit;
int main() {
    PhysicalNumber a(100, Unit::KM);
    PhysicalNumber b(2, Unit::HOUR);
    PhysicalNumber c(100, Unit::M);
    PhysicalNumber d(30, Unit::MIN);

    testcase
    .setname("Multiplication and division")
    .CHECK_OUTPUT(a/b, "50[km/hr]")    // 50 kilometers per hour
    .CHECK_OUTPUT(a*b, "200[km*hr]")
    .CHECK_OUTPUT(a*b*b, "400[km*hr*hr]")
    .CHECK_OUTPUT(a*c, "10000[km*m]")

    .setname("Adding compatible units")
    .CHECK_OUTPUT(a/b + a/d, "250[km/hr]")  // 50 km/hour + 200 km/hour
    .CHECK_OUTPUT(a/b - c/d, "49.8[km/hr]") // 50 km/hour - 200 m/hour

    .setname("Adding incompatible units")
    .CHECK_THROWS(a/b + a)  // a/b is [km/hour], a is [km]
    .CHECK_THROWS(a/b - a*b)  // a/b is [km/hour], a*b is [km*hour]

    // ADD MORE TESTS HERE
      .setname("...")
      .print(cout);
}
```

שאלה 6 [22 נק']

נתון קוד של משחק בול-פגיעה (כמו במטלה 4):

```
uint play(Chooser& chooser, Guesser& guesser, uint length, uint maxTurns) {
        const uint TECHNICAL_VICTORY_TO_GUESSER = 0;
        const uint TECHNICAL_VICTORY_TO_CHOOSER = maxTurns+1;

        string choice = chooser.choose(length);
        if (choice.length()!=length)       // Illegal choice
            return TECHNICAL_VICTORY_TO_GUESSER;
        guesser.startNewGame(length);  // tell the guesser that a new game starts
        uint indexOfTurn;
        for (indexOfTurn=0; indexOfTurn<maxTurns; ++indexOfTurn) {
            string guess = guesser.guess();
            if (guess.length()!=length)  // Illegal guess
                return TECHNICAL_VICTORY_TO_CHOOSER;
            if (guess==choice) {
                return indexOfTurn + 1;
            } else {
                auto reply = calculateBullAndPgia(choice, guess);
                guesser.learn(reply);
            }
        }
        return TECHNICAL_VICTORY_TO_CHOOSER;  // Guesser could not guess in time
    }
```

הערה: ניתן להניח שהקבצים Chooser.hpp, Guesser.hpp, calculate.hpp ושאר הקבצים ממטלה 4, כתובים בצורה נכונה – אין צורך לכתוב אותם מחדש.

א. כיתבו מחלקה בשם ShuffleChooser, הבוחרת תמיד מספר המורכב מהספרות 1 עד length, כל ספרה פעם אחת, בסדר אקראי (ניתן להניח ש length קטן או שווה 9). למשל, אם length=3 ייבחר באקראי אחד המספרים 123,132,213,231,312,321.

ב. כיתבו מחלקה בשם PermutationGuesser, המנסה לנחש את המספר שבחר ShuffleChooser ע"י בדיקת כל הפרמוטציות האפשריות מ-1 עד length . למשל, אם length=3 המנחש יבדוק את כל 6 המספרים הנ"ל לפי הסדר עד שימצא את התשובה הנכונה (או עד שייגמר הזמן).

לפניכם תוכנית ראשית לדוגמה:

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

#include "play.hpp"
#include "PermutationGuesser.hpp"
#include "ShuffleGuesser.hpp"

int main() {
    bullpgia::ShuffleChooser sc;
    bullpgia::PermutationGuesser pg;
    cout << bullpgia::play(sc, pg, 3, 100) << endl;
        // ShuffleChooser chooses a random permutation of 1,2,3
        // PermutationGuesser guesses all of them until the correct one is found.
    cout << bullpgia::play(sc, pg, 9, 100) << endl;
        // ShuffleChooser chooses a random permutation of 1,...,9.
        // PermutationGuesser guesses all of them until the correct one is found.
    return 0;
}
```

שימו לב: בסוף הבחינה מצורפים שלושה דפים מתוך התיעוד של שפת C++. הדפים נועדו להזכיר לכם פונקציות שלמדנו עליהן בהרצאה ועשויות לעזור לכם בפתרון השאלה.

שאלה 7 [22 נק']

כיתבו מחלקה בשם **binop**, שהבנאי שלה מקבל שלושה קלטים:

- שני דמויי-מיכלים (iterables) באותו אורך;
- פונקטור בינארי,

ובונה דמוי-מיכל (iterable) המתקבל מהפעלת הפעולה הבינארית על זוגות איברים תואמים בשני הקלטים. אם הקלטים אינם באותו אורך – ההתנהגות לא מוגדרת. לפניכם תוכנית לדוגמה.

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

#include "range.hpp"
#include "binop.hpp"

using namespace itertools;

int main() {
    vector<int> v1{10,20,30,40,50};
    list<double> L1{1.1, 2.2, 3.3, 4.4, 5.5};
    for (auto i: binop(v1, L1, [](int x,double y){return x+y;} ))
        cout << i << " "; // Prints 11.1 22.2 33.3 44.4 55.5
         // since 11.1 = 10 + 1.1, 22.2 = 20 + 2.2, etc.
    cout << endl;

    for (auto i: binop(v1, range(1,6), [](int x,int y){return x*y;} ))
        cout << i << " "; // Prints 10 40 90 160 250
         // since 10*1 = 10, 20*2 = 40, etc.
    cout << endl;
    return 0;
}
```

לפניכם התחלה של מימוש המחלקה בקובץ `binop.hpp`. השלימו את החסר.

```
namespace itertools {
    template<typename T1, typename T2, typename OP>
    class binop {
          ...
    };
}
```

שימו לב: אתם צריכים לכתוב רק את הקובץ `binop.hpp`. אתם **לא צריכים** לכתוב את הקובץ `range.hpp` – אפשר להניח שהוא כתוב ועובד נכון כמו במטלה. אין להניח הנחות נוספות.

שאלה 8 - מענקים

א [3 נק']. האם השתמשנו בבדיקות שלכם באחת המטלות? אם כן אנא ציינו באיזו מטלה בדיוק.

ב [2-4 נק']. האם זכיתם באחד משלושת המקומות הראשונים בתחרות בול-פגיעה? אם כן ציינו באיזה מקום.

נספח (לשאלה 6)

לפניכם שלושה דפים מתוך האתר **.cplusplus.com**

std::shuffle

```
template <class RandomAccessIterator, class URNG>
  void shuffle (RandomAccessIterator first, RandomAccessIterator last, URNG&& g);
```

Randomly rearrange elements in range using generator

Rearranges the elements in the range `[first,last)` randomly, using *g* as *uniform random number generator*.

The function swaps the value of each element with that of some other randomly picked element. The function determines the element picked by calling `g()`.

This function works with standard generators as those defined in <random>. To shuffle the elements of the range without such a generator, see random_shuffle instead.

The behavior of this function template is equivalent to:

```
template <class RandomAccessIterator, class URNG>
  void shuffle (RandomAccessIterator first, RandomAccessIterator last, URNG&& g)
{
  for (auto i=(last-first)-1; i>0; --i) {
    std::uniform_int_distribution<decltype(i)> d(0,i);
    swap (first[i], first[d(g)]);
  }
}
```

Parameters

first, last

Forward iterators to the initial and final positions of the sequence to be shuffled. The range used is `[first,last)`, which contains all the elements between *first* and *last*, including the element pointed by *first* but not the element pointed by *last*.
`ForwardIterator` shall point to a type for which `swap` is defined and swaps the value of its arguments.

g

A uniform random number generator, used as the source of randomness.
`URNG` shall be a *uniform random number generator*, such as one of the standard generator classes (see <random> for more information).

Return value

none

Example

Edit & Run

```
// shuffle algorithm example
#include <iostream>      // std::cout
#include <algorithm>     // std::shuffle
#include <array>         // std::array
#include <random>        // std::default_random_engine
#include <chrono>        // std::chrono::system_clock

int main () {
```

```
  std::array<int,5> foo {1,2,3,4,5};

  // obtain a time-based seed:
  unsigned seed =
std::chrono::system_clock::now().time_since_epoch().count();

  shuffle (foo.begin(), foo.end(), std::default_random_engine(seed));

  std::cout << "shuffled elements:";
  for (int& x: foo) std::cout << ' ' << x;
  std::cout << '\n';

  return 0;
}
```

Possible output:

```
shuffled elements: 3 1 4 2 5
```

Complexity

Linear in the distance between *first* and *last* minus one: Obtains random values and swaps elements.

Data races

The objects in the range `[first,last)` are modified.

Exceptions

Throws if any of the random number generations, the element swaps or the operations on iterators throws.
Note that invalid arguments cause *undefined behavior*.

---

std::next_permutation

default (1)
```
template <class BidirectionalIterator>
  bool next_permutation (BidirectionalIterator first,
                         BidirectionalIterator last);
```

custom (2)
```
template <class BidirectionalIterator, class Compare>
  bool next_permutation (BidirectionalIterator first,
                         BidirectionalIterator last, Compare comp);
```

Transform range to next permutation
Rearranges the elements in the range `[first,last)` into the next *lexicographically* *greater* permutation.

A *permutation* is each one of the `N!` possible arrangements the elements can take (where *N* is the number of elements in the range). Different permutations can be ordered according to how they compare *lexicographicaly* to each other; The first such-sorted possible permutation (the one that would compare *lexicographically smaller* to all other permutations) is the one which has all its elements sorted in ascending order, and the largest has all its elements sorted in descending order.

The comparisons of individual elements are performed using either `operator<` for the first version, or *comp* for the second.

If the function can determine the next higher permutation, it rearranges the elements as such and returns `true`. If that was not possible (because it is already at the largest possible permutation), it rearranges the elements according to the first permutation (sorted in ascending order) and returns `false`.

Parameters

first, last

Bidirectional iterators to the initial and final positions of the sequence. The range used is `[first,last)`, which contains all the elements between *first* and *last*, including the element pointed by *first* but not the element pointed by *last*.
`BidirectionalIterator` shall point to a type for which `swap` is properly defined.

comp

Binary function that accepts two arguments of the type pointed by `BidirectionalIterator`, and returns a value convertible to `bool`. The value returned indicates whether the first argument is considered to go before the second in the specific *strict weak ordering* it defines.
The function shall not modify any of its arguments.
This can either be a function pointer or a function object.

Return value

`true` if the function could rearrange the object as a lexicographicaly greater permutation.
Otherwise, the function returns `false` to indicate that the arrangement is not greater than the previous, but the lowest possible (sorted in ascending order).

Example

Edit & Run

```
// next_permutation example
#include <iostream>     // std::cout
#include <algorithm>    // std::next_permutation, std::sort

int main () {
  int myints[] = {1,2,3};

  std::sort (myints,myints+3);

  std::cout << "The 3! possible permutations with 3 elements:\n";
  do {
    std::cout << myints[0] << ' ' << myints[1] << ' ' << myints[2] << '\n';
  } while ( std::next_permutation(myints,myints+3) );

  std::cout << "After loop: " << myints[0] << ' ' << myints[1] << ' ' << myints[2] << '\n';

  return 0;
}
```

Output:

```
The 3! possible permutations with 3 elements:
1 2 3
1 3 2
2 1 3
2 3 1
3 1 2
3 2 1
After loop: 1 2 3
```

Complexity

Up to linear in half the distance between *first* and *last* (in terms of actual swaps).

Data races

The objects in the range `[first,last)` are modified.

Exceptions

Throws if any element swap throws or if any operation on an iterator throws.

Note that invalid arguments cause *undefined behavior*.

---

std::iota

```
template <class ForwardIterator, class T>
  void iota (ForwardIterator first, ForwardIterator last, T val);
```

Store increasing sequence

Assigns to every element in the range `[first,last)` successive values of *val*, as if incremented with `++val` after each element is written.

The behavior of this function template is equivalent to:

```
template <class ForwardIterator, class T>
  void iota (ForwardIterator first, ForwardIterator last, T val)
{
  while (first!=last) {
    *first = val;
    ++first;
    ++val;
  }
}
```

Parameters

first, last

Forward iterators to the initial and final positions of the sequence to be written. The range used is `[first,last)`, which contains all the elements between *first* and *last*, including the element pointed by *first* but not the element pointed by *last*.

val

Initial value for the accumulator.

Return value

none

Example

```
// iota example
#include <iostream>      // std::cout
#include <numeric>       // std::iota

int main () {
  int numbers[10];
```

Edit & Run

```

  std::iota (numbers,numbers+10,100);

  std::cout << "numbers:";
  for (int& i:numbers) std::cout << ' ' << i;
  std::cout << '\n';

  return 0;
}
```

Output:

```
numbers: 100 101 102 103 104 105 106 107 108 109
```

Complexity

Linear in the distance between *first* and *last* (increments and assignments).

Data races

The elements in the range `[first,last)` are modified (each element is modified exactly once).

Exceptions

Throws if any of the assignments or increments throws.

Note that invalid arguments cause *undefined behavior*.