פקולטה: מדעי הטבע. מחלקה: מדעי המחשב
שם הקורס: תכנות מערכות ב .
קוד הקורס: 7020411 **כל הקבוצות**
מועד: **ב** סמסטר: **ב** שנה: ה'תשפ"ד
תאריך הבחינה: י באב תשפד 14.8.24
משך הבחינה: שעתיים וחצי – 150 דקות

מרצה אחראי : ד"ר אראל סגל-הלוי
מרצים : מר מור בסן, ד"ר אלינה (בננסון) אופלינסקי
מתרגלים: מר ליאור ברייטמן, מר עמית רזינובסקי, מר בנימין סאלדמן, מר יהונתן עמוסי

חל איסור להשתמש בחומר עזר \ מחשבון \ מחשב.

יש לענות על כל השאלות במחברת הבחינה בלבד, בכתב ברור וקריא!

במבחן 5 שאלות, על 8 עמודים, יש לענות רק במחברת ולא על הטופס, יש לענות על כל השאלות

אנא קראו היטב את כלל ההנחיות והשאלות לפני כתיבת התשובות

יינתן מענק של 2 נקודות על כתיבה מסודרת בהתאם לקיום כלל הסעיפים הבאים:

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

יש לענות תשובות מלאות, להסביר כל תשובה בפירוט, ולכתוב תיעוד לקוד ושמות משמעותיים.
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, ציינו והסבירו מה הבנתם ולמה התכוונתם, והמשיכו לשאלה הבאה. אל "תיתקעו" בשאלה אחת.

בהצלחה!!!

שאלה 0 [50 נק']

[ציון על מטלות והצגות\נוכחות במעבדות]

כפי שנכתב בסילבוס : ציון זה יתווסף רק בתנאי של קבלת 30 נקודות או יותר במבחן

שאלה 1 [10 נק'] – בקיאות (2 נק' לסעיף)

בסעיפים אלו יש לענות בצורה קצרה אך ברורה עם דוגמאות קוד.

1. מהי תבנית – Template – בשפת ++C . אילו סוגי תבניות ניתן ליצור מלבד פונקציות? תנו דוגמא קצרה להסבר. באיזה סוג קובץ יש לכתוב קוד זה, ומדוע?

מנגנון המאפשר כתיבת קוד גנרי שעובד עם סוגי נתונים שונים. במקום לכתוב פונקציה או מחלקה עבור כל סוג נתונים בנפרד, ניתן לכתוב קוד גנרי בעזרת תבנית, **והקומפיילר** ייצר את הקוד המתאים עבור כל סוג נתונים בעת הצורך. סוגים:

Function Templates פונקציות גנריות.

מחלקות Class Templates מחלקות גנריות. וראינו גם משתנים גנריים

קוד תבניות בדרך כלל נכתב בקובצי כותרות (`hpp.` או `h.`). הסיבה לכך היא שהתבניות אינן יוצרות קוד בפועל בזמן ההגדרה שלהן, אלא רק בזמן השימוש בהן, ולכן יש להן גישה מלאה לקוד כדי לאפשר לקומפיילר לייצר את המימוש המתאים עבור סוג הנתונים המבוקש.

2. האם שפת ++C תומכת בשימוש\שילוב קוד משפת C?

כן

ציינו 3 הבדלים בין שפת ++C לשפת C (מבחינת עקרונות ולא רק תחבירית).

תכנות מונחה עצמים(OOP)
העמסת פונקציות ואופרטורים
ניהול זיכרון
חריגות(Exceptions)
תבניות(Templates)
מנגנון שמות(Namespaces)
בנאים\מפשקים
פולימורפיזם
פונקציותInline
חברויות(Friendship)

3. הסבירו את מה זה רשימת אתחול initialization list וציינו 2 יתרונות בשימוש בה. אם מחלקה מסוימת מכילה בתוכה מחלקה אחרת (composition) , איך ישפיע הקוד ב initialization list על אופן או סדר בניית המחלקות במחלקה?

מאפשרת אתחול של חברי מחלקה ישירות במהלך קריאת הבנאי, לפני שהגוף של הבנאי מתבצע. הדבר חשוב במיוחד כאשר מדובר בחברי מחלקה קבועים (`const`) או כאשר החברים הם אובייקטים של מחלקות אחרות שאין להן בנאי ברירת מחדל.

ביצועים: אתחול חברי המחלקה ברשימת אתחול מונע קריאות כפולות של בנאים ומבטיח יעילות גבוהה יותר בהשוואה לאתחול בתוך גוף הבנאי.
אתחול שדות קבועים: שדות קבועים (const) יכולים להיות מאותחלים רק ברשימת אתחול ולא בתוך גוף הבנאי.

כאשר מחלקה אחת מכילה מחלקות אחרות כשדות, אתחול המחלקות הללו מתבצע לפי הסדר שבו הן מוגדרות במחלקה. שימוש ברשימת אתחול מבטיח שהמחלקות הפנימיות יאותחלו בצורה נכונה לפי הסדר הנדרש.

4. מהו unique_ptr ומה היתרון בשימוש בו ? כתבו קוד דוגמא למימוש (חתימות וקוד מינימלי) להשלמת ההסבר .

מצביע חכם ב-C++ שמנהל זיכרון בצורה אוטומטית. הוא מחזיק באחראיות ייחודית על אובייקט דינמי, כלומר, רק מצביע אחד יכול להחזיק בו בכל זמן נתון. ברגע שהמצביע יוצא מה SCOPE הזיכרון שאליו הוא מצביע משוחרר אוטומטית. המונע זליגת זיכרון ומבטל את הצורך בקריאות ל-`delete` ידני.

```
#include <iostream>

template<typename T>
class MyUniquePtr {
private:
    T* ptr;

public:
    // Constructor
      MyUniquePtr(T* p ) : ptr(p) {}

    // Destructor
    ~MyUniquePtr() {
        delete ptr;
    }

    // Disable copy constructor and copy assignment
    MyUniquePtr(const MyUniquePtr&) = delete;
    MyUniquePtr& operator=(const MyUniquePtr&) = delete;
    }

    // Dereference operator
    T& operator*() const { return *ptr; }

    // Arrow operator
    T* operator->() const { return ptr; }
};

class MyClass {...};

int main() {
    MyUniquePtr<MyClass> ptr(new MyClass());
```

```
        return 0;
{
```

5. מהו functor (function object) ? תנו דוגמא להגדרה ולשימוש בו בשילוב ספריית STL.

אובייקט במחלקה ב-C++ שמתנהג כמו פונקציה באמצעות הגדרת האופרטור `()` במחלקה.

```
#include <iostream>
#include <vector>
#include <algorithm>

class Compare {
public:
    bool operator()(int a, int b) const {
        return a < b;
    }
};

int main() {
    std::vector<int> nums = {4, 2, 5, 1, 3};
    std::sort(nums.begin(), nums.end(), Compare());
    for (int n : nums) std::cout << n << " ";
    return 0;
}
```

שאלה 2 [10 נק']

נתון הקוד הבא

```
#include <iostream>
#include <stdexcept>

template<typename T, std::size_t K>
class K_vector {
private:
    T* vec;
    std::size_t currentSize;
public:
    K_vector() : vec(new T[K]), currentSize(0) {}

    ~K_vector() {
        delete[] vec;
    }
    void push_back(const T& element) {
        if (currentSize >= K) {
            throw std::out_of_range("Vector is full");
        }
        vec[currentSize++] = element;
    }
};

template <typename T, std::size_t K>
void K_func(K_vector<T,K> vect)
{
    // some code here
}

int main()
{
    K_vector<int, 5> container;
    for (int i = 0; i < 5; i++)
        container.push_back(i);
    K_func(container);
}
```

סעיף א [4 נק]

מצאו את השגיאה שקיימת בקוד הנתון, באיזה שלב היא תופיע, הוסיפו קוד שמתקן את הבעיה.

השגיאה היא העברת אובייקט BY VALUE ובכל מקרה חסר בנאי מעתיק ואופרטור השמה
קוד מלא בסוף השאלה

סעיף ב [6 נק]

נתון המימוש של פונקציה K_func. אילו שגיאות קיימות בקוד זה? הוסיפו קוד למחלקה K_vector שיתקן את **כל** השגיאות.

```
template <typename T, std::size_t K>
void K_func(K_vector<T,K> vect)
```

```
{
    vect << 2; // cyclic shift 12340 -> 34012
    std::cout << vect;
}
```

```
#include <iostream>
#include <stdexcept>

template<typename T, std::size_t K>
class K_vector {
private:
    T* vec;
    std::size_t currentSize;

public:
    K_vector() : vec(new T[K]), currentSize(0) {}

    // Copy Constructor
    K_vector(const K_vector& other) : vec(new T[K]), currentSize(other.current-
Size) {
        for (std::size_t i = 0; i < currentSize; ++i) {
            vec[i] = other.vec[i];
        }
    }

    // Assignment Operator
    K_vector& operator=(const K_vector& other) {
        if (this == &other) {
            return *this;
        }
        delete[] vec;

        vec = new T[K];
        currentSize = other.currentSize;
        for (std::size_t i = 0; i < currentSize; ++i) {
            vec[i] = other.vec[i];
        }

        return *this;
    }

    ~K_vector() {
        delete[] vec;
    }

    void push_back(const T& element) {
        if (currentSize >= K) {
            throw std::out_of_range("Vector is full");
        }
        vec[currentSize++] = element;
    }
    // Operator for shifting.
      K_vector& operator<<(int shift) {
         if (currentSize > 0) {
             shift = shift % currentSize;
             T* temp = new T[currentSize];

             for (std::size_t i = 0; i < currentSize; ++i) {
                 temp[i] = vec[(i + shift) % currentSize];
             }
```

```
            for (std::size_t i = 0; i < currentSize; ++i) {
                vec[i] = temp[i];
            }

            delete[] temp;
        }
        return *this;
    }

    // Output operator
    friend std::ostream& operator<<(std::ostream& os, const K_vector& kv)
{
        for (std::size_t i = 0; i < kv.currentSize; ++i) {
            os << kv.vec[i] << " ";
        }
        return os;
    }
};

template <typename T, std::size_t K>
void K_func(K_vector<T,K> vect)
{
 vect << 2; // cyclic shift 12340 -> 34012
 std::cout << vect;
}

int main() {
    /**
      * k_func הבעיה היא שאנחנו מעבירים העתק לפונקציה
      * בלי מימוש של אופרטור השמה ובנאי מעתיק (למרות שיצרנו מפרק)
      * נתקן את הקוד על ידי זה שנוסיף בנאי מעתיק ואופרטור השמה למחלקה (כלל השלושה).
      *
      *  סעיף ב' – צריך להוסיף למחלקה שני אופרטורים, הראשון מקבל מספר ומבצע הזזה ציקלית של
המערך והשני אופרטור פלט. הקוד כתוב כבר בתוך המחלקה.
      */
    K_vector<int, 5> container;
    for (int i = 0; i < 5; i++) {
        container.push_back(i);
    }
    std::cout<<container<<std::endl;
    K_func(container);
}
```

שאלה 3 [10 נק']

נתונות המחלקות הבאות :

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

class A {
public:
    virtual void func1() { cout << "A::func1" << endl; }
    void func2() const { cout << "A::func2" << endl; }
    void func3() { cout << "A::func3" << endl; }
};
```

```
class B : public A {
public:
    virtual void func1() { cout << "B::func1" << endl; }
    void func2() { cout << "B::func2" << endl; }
    void func3() { cout << "B::func3" << endl; }
    virtual void func4() { cout << "B::func4" << endl; }
};

class C : public B {
public:
    void func1() const { cout << "C::func1" << endl; }
    void func3() { cout << "C::func3" << endl; }
    virtual void func4() { cout << "C::func4" << endl; }
};

class D : public C {
public:
    void func1() { cout << "D::func1" << endl; }
    void func2() { cout << "D::func2" << endl; }
    void func3() { cout << "D::func3" << endl; }
    void func4() { cout << "D::func4" << endl; }
};

int main() {
    B b; C c; D d;
    A* pa = &b;
    B* pb = &c;
    A* pc = &d;
    pa->func2();
    pc->func3();
    pb->func3();
    pb->func4();
    return 0;
}
```

א . [3 נק'] ציינו עבור כל מחלקה את טבלת vtable אשר נשמרת עבורה

יש לציין מהי הפונקציה הרלוונטית (מימוש) לכל למחלקה –

לדוגמא מימוש של פונקציית foo במחלקת base - יש לרשום base::foo()

3.1

A:

func1: A::func1

B:

func1: B::func1

func4: B::func4

C:

func1: C::func1

func4: C::func3

D:

func1: D::func1

func4: D::func4

ב . [2 נק'] כתבו מה יודפס בעת הרצת הקוד.

A:: func2

A:: func3

B:: func3

C:: func4

ג. [5 נק'] ענו על כל שאלות הבאות:

- מהי פונקציה וירטואלית טהורה (pure virtual)?
- כתבו דוגמא לקוד עם פונקציה וירטואלית טהורה.
- מה מיוחד במחלקה שמכילה פונקציה pure virtual.
- נתונה מחלקת Base עם מספר פונקציות שונות וביניהם פונקציה וירטואלית טהורה.

כעת נוסיף מחלקה חדשה Derived, אשר יורשת ממחלקת Base ללא שום מימושים ותוספות נוספות (ראו הקוד בשורות הבאות). נוסיף את הקוד הבא עם התוכנית הראשית main. האם הקוד יעבור הידור (קימפול) ? (הניחו כי אין שגיאה או בעיות במחלקת Base)

```
class Derived : public Base {
};

int main()
{
    Derived d;
    return 0;
}
```

3.3

א. פונקציה וירטואלית טהורה (pure virtual function) היא פונקציה במחלקה בסיסית שמוגדרת כך שאין לה מימוש במחלקה הבסיסית, והיא מחייבת את המחלקות הנגזרות לספק מימוש עבור הפונקציה. פונקציה כזו הופכת את המחלקה הבסיסית למחלקה אבסטרקטית, כלומר מחלקה שלא ניתן ליצור אובייקטים ממנה ישירות.
אם מחלקה נגזרת לא תממש את הפונקציה הוירטואלית הטהורה, גם היא תהפוך למחלקה אבסטרקטית, ולא ניתן יהיה ליצור אובייקטים ממנה.

ב.

```
class Base {
public:
   virtual void pureVirtualFunction() = 0;
};

class Derived : public Base {
public:
   void pureVirtualFunction() override {
מימוש של הפונקציה במחלקה הנגזרת //
      cout << "Derived implementation of pureVirtualFunction" << endl;
   }
};
```

ג. מדובר במחלקה אבסטרקטית ולא ניתן ליצור אובייקטים ממחלקה אבסטרקטית ומחייבת מימוש במחלקות נגזרות.

ד. הקוד לא יעבור הידור מכיוון שהמחלקה Derivedלא מממשת את הפונקציה הווירטואלית הטהורה pureVirtualFunction. מחלקה המכילה פונקציה וירטואלית טהורה היא מחלקה אבסטרקטית, וכך גם כל מחלקה שיורשת ממנה מבלי לממש את כל הפונקציות הווירטואליות הטהורות תהפוך גם היא למחלקה אבסטרקטית.
כיוון ש Derived- היא מחלקה אבסטרקטית, לא ניתן ליצור ממנה אובייקט ישירות, ולכן השורה Derived d; ב main- תגרום לשגיאת קומפילציה.

שאלה 4 [10 נק']

נתונה מחלקה הבאה

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

class Graph {
private:
    int numVertices;
    vector<list<int>> adjLists;

public:
    Graph(int vertices) : numVertices(vertices), adjLists(vertices) {}
```

```
    void addEdge(int src, int dest) {
        if (src >= numVertices || dest >= numVertices
              || src < 0 || dest < 0) {
            throw invalid_argument("Invalid vertex number");
        }
        adjLists[src].push_back(dest);
        adjLists[dest].push_back(src);
    }

    void printGraph() const {
        for (int i = 0; i < numVertices; ++i) {
            cout << "Vertex " << i << ":";
            for (int adj : adjLists[i]) {
                cout << " " << adj;
            }
            cout << endl;
        }  }
};
```

הוסיפו למחלקה Graph איטרטור Iterator שמאפשר לעבור על **כל הקשתות** בגרף. על האיטרטור לכלול את האופרטורים הדרושים -pre/post ++ , אופרטור השוואה ואי-שוויון, ואופרטורי גישה (* ו ->)
שימו לב: לרשותכם מחלקה סטנדרטית **std::pair< typename T1,typename T2>** כמו שנלמד בהרצאות.
הוסיפו פונקציות מתאימות במחלקה Graph כדי לתמוך בקוד הבא:

```
int main() {
    Graph g(5);
    g.addEdge(0, 1);    g.addEdge(0, 4);  g.addEdge(1, 2);  g.addEdge(1, 3);
    g.addEdge(1, 4);    g.addEdge(2, 3);  g.addEdge(3, 4);
    g.printGraph();  // Prints the initial graph

    cout << "Edges in the graph: ";
    for (auto it = g.begin(); it != g.end(); ++it) {
        auto edge = *it;
        cout << "(" << edge.first << ", " << edge.second << ") ";
    }
    cout << endl;
    return 0; }
```

פתרון :

```
#include <iostream>
#include <vector>
#include <list>
#include <stdexcept>
#include <utility>
using namespace std;
class Graph {
   using Vertex = int;
   using AdjList = std::vector<std::list<Vertex>>;
   using Edge = std::pair<Vertex, Vertex>;
   private:
    int numVertices;
    vector<list<int>> adjLists;
   public:
    Graph(int vertices) : numVertices(vertices), adjLists(vertices) {}
    void addEdge(int src, int dest) {
      if (src >= numVertices || dest >= numVertices || src < 0 || dest < 0) {
         throw invalid_argument("Invalid vertex number");
      }
      adjLists[src].push_back(dest);
      adjLists[dest].push_back(src);
      }
   void printGraph() const {
      for (int i = 0; i < numVertices; ++i) {
         cout << "Vertex " << i << ":";
            for (int adj : adjLists[i]) {
               cout << " " << adj;
   }
   cout << endl;
 } }

   class EdgeIterator {
   public:
      EdgeIterator(const AdjList& adjList, bool isEnd = false)
         : adjList(adjList), outerIndex(0) {
         if (!isEnd) {
            outerIndex = 0;
            if (!adjList.empty()) {
               innerIt = adjList[outerIndex].begin();
               advanceToNextValid();
            }
         } else {
```

```
            outerIndex = adjList.size();
        }
    }

    bool operator!=(const EdgeIterator& other) const {
        return outerIndex != other.outerIndex;
    }

    EdgeIterator& operator++() {
        ++innerIt;
        advanceToNextValid();
        return *this;
    }

    Edge operator*() const {
        return { static_cast<Vertex>(outerIndex), *innerIt };
    }

    const Edge* operator->() const {
        currentEdge = { static_cast<Vertex>(outerIndex), *innerIt };
        return &currentEdge;
    }

private:
    void advanceToNextValid() {
        while (outerIndex < adjList.size() && innerIt == adjList[outerIndex].end()) {
            ++outerIndex;
            if (outerIndex < adjList.size()) {
                innerIt = adjList[outerIndex].begin();
            }
        }
    }

    const AdjList& adjList;
    size_t outerIndex;
    typename std::list<Vertex>::const_iterator innerIt;
    mutable Edge currentEdge; // To store the current edge for the -> operator
};

EdgeIterator begin() const{
    return EdgeIterator(adjLists);
}
```

```
    EdgeIterator end() const{
        return EdgeIterator(adjLists,true);
    }
};

int main() {
    Graph g(5);
    g.addEdge(0, 1); g.addEdge(0, 4); g.addEdge(1, 2); g.addEdge(1, 3);
    g.addEdge(1, 4); g.addEdge(2, 3); g.addEdge(3, 4);
    g.printGraph(); // Prints the initial graph

     cout << "Edges in the graph: ";
    for (auto it = g.begin(); it != g.end(); ++it) {
        auto edge = *it;
        cout << "(" << edge.first << ", " << edge.second << ") ";
    }
     cout << endl;
 return 0;

}
```

שאלה 5 – [10 נק']

א. [3 נק']

הגדירו מחלקה כללית (Template) בשם MyPair אשר תכיל שני משתנים **פרטיים** מאותו סוג T .
בנוסף הגדירו פונקציה בשם isEqual, הפונקציה תחזיר true אם שני המשתנים בתוך ה- pair הינם שווים, אחרת false. דוגמא לשימוש :

```
int main() {
    MyPair<int> intPair1(5, 5);
    MyPair<int> intPair2(3, 7);
    MyPair<std::string> strPair("hello", "world");

    std::cout << "intPair1 are equal: " << (isEqual(intPair1) ? "true" :
"false") << std::endl; // true

    std::cout << "intPair2 are equal: " << (isEqual(intPair2) ? "true" :
"false") << std::endl; //false

    std::cout << "strPair are equal: " << (isEqual(strPair) ? "true" : "false")
<< std::endl; //false

    return 0;

}
```

הפתרון ביחד עם הסעיף הבא בהמשך

ב. [3 נק']

הוסיפו קוד למחלקת **MyPair** כך שיהיה ניתן **להשוות** בין 2 אוביקטים מסוג **MyPair**
וכן לאפשר הדפסה נוחה. לדוגמא עבור השימוש הבא : (בהמשך לקוד הקודם בסעיף א')

```
std::cout << "p1 < p2: " << (p1 == p2 ? "true" : "false") << std::endl;

std::cout << "p1: " << p1 << std::endl;
```

```
#include <iostream>
#include <string>

template<typename T>
class MyPair {
private:
    T first;
    T second;

public:

    MyPair(T a, T b) : first(a), second(b) {}

    bool isEqual() const {
        return first == second;
    }

    bool operator==(const MyPair<T>& other) const {
        return first == other.first && second == other.second;
    }


    friend std::ostream& operator<<(std::ostream& os, const MyPair<T>& pair) {
        os << "(" << pair.first << ", " << pair.second << ")";
        return os;
    }
};


template<typename T>
bool isEqual(const MyPair<T>& pair) {
    return pair.isEqual();

}
```

5.ג [4 נק']

בקוד הבא נפלו מספר שגיאות\בעיות. מצאו את כל השגיאות \ בעיות והציעו תיקונים מתאימים (**ללא הוספת**

שורות חדשות) כך שהקוד יוכל לעבור הידור ולרוץ.
שימו לב : חובה לציין במחברת את מספר שורה בה הנכם מעוניינים להכניס תיקון.

```
1  #include <iostream>
2  #include <string>
3
4  class Person {
5      std::string name;
6  public:
7      Person(const std::string& name) : name(name) {}
8      virtual ~Person() {}
9      void printInfo() {
10         std::cout << "printInfo Person";
11     }
12 };
13
14 class Student : public Person {
15 public:
16     Student(const std::string& name, int grade) : Person(name), grade(grade) {}
17     void printInfo() {
18         std::cout << "printInfo Student";
19     }
20     void printGrade() {
21         std::cout << "Grade:" << grade;
22     }
23 protected:
24     int grade;
25 };
26
27 int main() {
28     Person person1 = new Student("Shay", 85);
29     person1->printInfo();
30     person1->printGrade();
31
32     return 0;
33 }
```

שורה 11: הפונקציה printInfo במחלקת Person אינה וירטואלית

שורה 20: הפונקציה printInfo במחלקת Student אינה מוגדרת עם override,

שורה 28: משתנה צריך להיות מצביע

שורה 29: הפונקציה printGrade לא קיימת במחלקת Person

שורה 31: שחרור זיכרון חסר

שאלה 6 בונוס [2 נק']

יינתן מענק של 2 נקודות על כתיבה מסודרת בהתאם לקיום **כלל** הסעיפים הבאים:

- כלל השאלות פתורות במחברת הבחינה לפי הסדר שבשאלון;

- כל שאלה מתחילה בעמוד נפרד;
- הכתב ברור וקריא, ללא מחיקות \ קשקושים\ חיצים \ וטקסט מיותר.