

**מבחן מועד א' בקורס: "תכנות מונחה עצמים"
סמסטר אביב תשס"ד**

- משך המבחן: שעתיים וחצי.
- מותר חומר עזר כתוב בלבד.
- ליד כל סעיף מצוין מספר הנקודות.
- את התשובות יש לכתוב על גבי טופס הבחינה בלבד. ניתן להעזר במחברת כטייטה.
- הבוחן כולל שלושה חלקים. יש לענות על כל השאלות.
- חלק מהשאלות דורשות בחירת תשובה אחת מתאימה מבין תשובות מוצעות.
- חלק מהשאלות דורשות השלמת מלל, הסבר או הבהרה.
- חלק מהשאלות דורשות השלמת קוד.
- הניקוד לכל שאלה מופיע בצידה.

ניקוד:

שאלה 1	/ 3	חלק א (39 נק')
שאלה 2	/ 3	
שאלה 3	/ 3	
שאלה 4	/ 3	
שאלה 5	/ 3	
שאלה 6	/ 3	
שאלה 7	/ 3	
שאלה 8	/ 3	
שאלה 9	/ 3	
שאלה 10	/ 3	
שאלה 11	/ 3	
שאלה 12	/ 3	
שאלה 13	/ 3	
שאלה 14	/ 10	חלק ב (30 נק')
שאלה 15	/ 10	
שאלה 16	/ 10	
שאלה 17.1	/ 5	חלק ג (36 נק')
שאלה 17.2	/ 10	
שאלה 17.3.1	/ 3	
שאלה 17.3.2	/ 3	
שאלה 17.3.3	/ 3	
שאלה 17.3.4	/ 3	
שאלה 17.3.5	/ 3	
שאלה 17.3.6	/ 3	
שאלה 17.3.7	/ 3	
סה"כ	/105	



1. מאיזה בעייתיות סובל קטע הקוד הבא:

```
1. #include <stdio.h>
2. class X { ... //some implementation };
3. void main() {
4.     const X* px = new X("arg", 8);
5.     free((void*)px);
6. }
```

א. שגיאת הידור (קומפילציה) מאחר ולא ניתן לשחרר זיכרון דינאמי ע"י free (שורה 5) אלא ע"י delete.

ב. שגיאת הידור (קומפילציה) מאחר ולא ניתן אובייקט רגיל למצביע שהוא const (שורה 4).

ג. שחרור זיכרון לקוי מאחר ולא יופעל destructor לאובייקט.

ד. הקצאת זיכרון לקויה מאחר ולא ניתן להקצות אובייקט X עם פרמטרים.

2. נתונות שלושת הפונקציות הגלובליות הבאות המחשבות ציון לסטודנט במבחן ב- oop:

```
(1) int oopTest ( long studentID ) { return (50+rand()%10); }
(2) const char* oopTest ( char* const studentID ) { return "50"; }
(3) long oopTest ( long studentID, long copyingFrom=0) { return (30+rand()%10); }
```

א. הפונקציות מוגדרות היטב ועוברות הידור ללא כל בעייה.

ב. הקוד לא יעבור הידור בגלל שכפול של הגדרה (multiple definition) של הפונקציות.

ג. הקוד לא יעבור הידור בגלל הערך המוחזר באחת הפונקציות שאינו תואם את הגדרתה.

ד. הקוד לא יעבור הידור בגלל שאין שימוש בערכים המועברים לפונקציות.

3. היכן לא כדאי אף פעם להגדיר אובייקט זמני מטיפוס X : X temp; ?

א. בבנאי כלשהו של המחלקה X.

ב. במפרק של המחלקה X.

ג. באופרטור ההשמה (=).

ד. תשובות (א) ו- (ב) נכונות.

4. במידה וכל הבנאים (constructors) של מחלקה X מוגדרים בחלק הפרטי (private) של המחלקה,

א. לא ניתן להגדיר אובייקט מטיפוס X בשום צורה שהיא.

ב. רק מתודה סטטית של המחלקה X יכולה ליצור אובייקט מטיפוס X.

ג. רק מתודה שאינה סטטית של אובייקט מטיפוס X יכולה ליצור אובייקט נוסף מאותו טיפוס (X).

ד. אף תשובה אינה נכונה.

5. איזו מתבניות (templates) הבאות אינה תקינה (לא בהכרח תעבור הידור)?

א. `template <class T, class S > T& f(int& t1, S t2) { return t1; }`

ב. `template <class T, class S=int> class L{ int f(T& t, S& s){ return 0; } };`

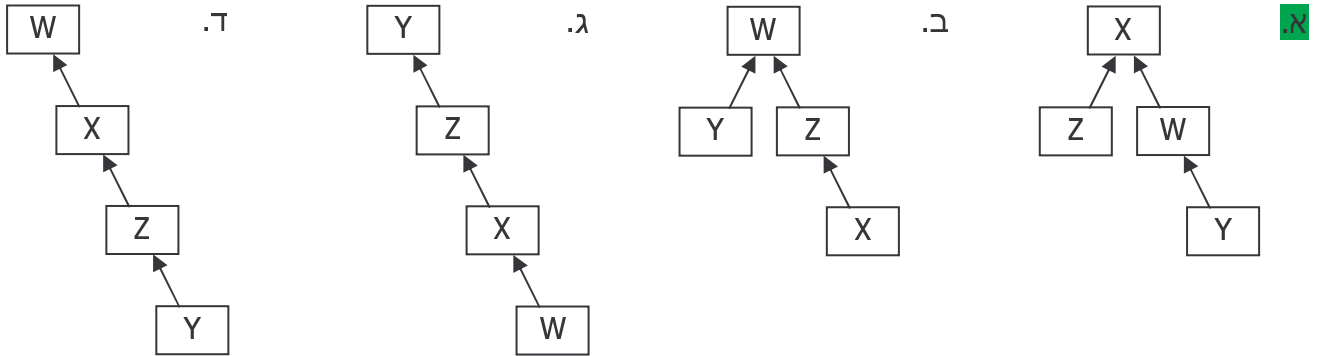
ג. `template <class T=int, class S > class V{ public: V(S&); };`

ד. `template <class T> class M{ T** t; };`



6. נתונות 4 מחלקות - X, Y, Z ו- W אשר יש ביניהן יחסי ירושה, ואף ייתכנו יחסי הכלה. כאשר מפורק אובייקט מטיפוס Y נקראים המפרקים הבאים של המחלקות בסדר הבא משמאל לימין:

$\sim X()$, $\sim W()$, $\sim X()$, $\sim Z()$, $\sim Y()$
 מה מהגרפים הבאים יכול לתאר את יחסי הירושה בין המחלקות הנ"ל?



7. אילו אופרטורים צריכים להיות מוגדרים על מחלקה לטיפול במספרים שלמים גדולים בשם BigInt כדי שהקוד הבא יעבור הידור (קומפילציה):

```
BigInt b(1234);
const char* intAsStr = b + 15;
```

- א. operator int -1 operator const char*
- ב. operator+ -1 operator const char*
- ג. operator= -1 operator+
- ד. operator int -1 operator+

8. להלן מימוש של תבנית שמטרתה לבצע חישוב מכסימום בין שלושה איברים מאותו טיפוס:

```
template <class T>
T MyMax(T left, T mid, T right)
{
    int big = (left>mid) ? left : mid;
    return ((big>right) ? big : right);
}
```

- א. תתקבל שגיאת הידור מאחר ולא ניתן לדעת האם הוגדר על הטיפוס המבוקש >operator.
- ב. התבנית תחזיר את הערך 4.7 כאשר נקרא לה באופן הבא: MyMax(5.9, 5.0, 4.7).
- ג. התבנית תחזיר את הערך 5.0 כאשר נקרא לה באופן הבא: MyMax(5.9, 5.0, 4.7).
- ד. התבנית תחזיר את הערך 5.9 כאשר נקרא לה באופן הבא: MyMax(5.9, 5.0, 4.7).

9. איטרטור הינו:

- א. סוג של מצביע לאיבר במכולה (container).
- ב. מחלקה שמוגדרות עליה פעולות ++, --, *, >, ->.
- ג. מחלקה בעלת ידע וגישה לייצוג הפנימי של המכולה בה היא מוגדרת.
- ד. מחלקה שבעזרתה ניתן לפנות לשדות בתוך המכולה בה היא מוגדרת.



10. נניח כי הגדרנו רשימה מקושרת של stl באופן הבא:
`list<list<int*>*> mylist;`
 כיצד נגדיר איטרטור העובר על איברי רשימה זו?

- א. `mylist::iterator it;`
- ב. `list<int*>::iterator it;`
- ג. `list<list<int*>*>::iterator it;`
- ד. `list<int*>*>::iterator it;`

11. נתונה הפונקציה הבאה:

```

1. int& f (int& i) {
2.     switch(i){
3.         case(0) : { int x = 6;         return x; }
4.         case(1) : { static int y = 7;   return y; }
5.     }
6.     int* z = new int(8);
7.     return *z;
8. }
```

- א. בשורה (3) יש שגיאת הידור (קומפילציה) מאחר ומוחזר ערך לוקלי.
- ב. בשורה (4) יש שגיאת הידור מאחר ומוחזר ערך סטטי.
- ג. בשורה (7) יש שגיאת הידור מאחר ומוחזר ערך המוקצה דינאמית.
- ד. אין שגיאות הידור עבור הפונקציה הנ"ל, אך תתכן אזהרה (warning) כלשהי.

12. מה תהיה ההדפסה בסוף קטע הקוד הבא:

```

inline int f (int& x) { x++; return x; }

void main(){
    int x, y=100, *z=&y;
    x=f(y);
    *z=f(x);
    y=f(*z);
    cout << x << ' ' << y << ' ' << *z ;
}
```

- א. 102,102,102
- ב. 103,103,103
- ג. 102,103,103
- ד. 102,103,102

13. תהי B מחלקת יסוד כלשהי המגדירה בנאי יחיד המקבל כפרמטר int. תהי D מחלקה הנגזרת ממנה ומגדירה בנאי יחיד המקבל כפרמטר String. כיצד יכול להיות מוגדר הבנאי של מחלקה D?

- א. `D::D(String s) { B(8); }`
- ב. `D::D(String s) : B(8) {}`
- ג. `D::D(String s) : B=8 {}`
- ד. `D::D(String s) {}`



חלק ב'

בחלק זה 3 שאלות קצרות שוות ניקוד (10 נקודות כל אחת). יש לענות במקום הנדרש בדף בחינה זה.

שאלה 14:

נתונה המחלקה הבאה:

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

class MyList : public list<int> {
public:
    void join();
};
```

עליכם לממש את המתודה join אשר מקבצת ערכים דומים יחדיו למופע הראשון שלהם ברשימה. לדוגמא:

12→2→5→2→11→3→2→11→7
12→2→5→11→3→7

הרשימה המקורית:
after call to join();

עליכם לממש את המתודה בהמשך להגדרת המחלקה. אין לכתוב פונקציות עזר, אך ניתן לקרוא לפונקציות/מתודות קיימות.

```
void MyList::join() {
{
    while(!empty())
    {
        int i = front();
        pop_front();
        iterator it;
        while((it=find(begin(), end(), i)) != end())
            erase(it);
        temp.push_back(i);
    }
    *this = temp;
}
```

שאלה 15:

לפניכם קטע קוד ובו 3 שגיאות הידור (קומפילציה). מהן השגיאות וכיצד ניתן לתקנן?

```
1. #include <string>
2. #include <iostream>
3. using namespace std;

4. class File {
5.     inline virtual void print() const { cout << "File name is: " << name; }
6. protected:
7.     string name;
8. public:
9.     File(string &s) : name(s) { }
10.    string& getName() const { return name; }
11. };

12. class Dir : public File {
13.     unsigned int dirsize;
14. public:
15.     inline virtual void print() const { File::print(); cout << "Dir size is: " << dirsize; }
16.     Dir(int size) : dirsize(size) { }
17. };
```

שורה: 11 שגיאה: לא ניתן לפנות למתודה File::print מאחר והיא בחלק הפרטי

תיקון: להעבירה במחלקה File לחלק ה-protected

שורה: 8 שגיאה: מחזירים ייחוס לא קבוע במתודה קבועה

תיקון: להחזיר const string& או string או לשנות את המתודה להיות לא קבועה

שורה: 12 שגיאה: לא מועבר ערך לאתחול File ואין ל-File בנאי ברירת מחדל

תיקון: להגדיר בנאי ברירת מחדל ל-File או לקרוא בשורת הפרמטרים לבנאי הקיים של File

מה יהיה הפלט של התוכנית הבאה?

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

class Vehicle {
protected:
    unsigned km;
public:
    Vehicle(unsigned k=0) : km(k) { cout<<"hello\n"; }
    ~Vehicle() { cout<<"bye bye\n"; }
    virtual void drive(unsigned k) { km+=k; cout<<km<<endl; }
    void drive() { km++; cout<<km<<endl; }
};

class Car : public Vehicle {
public:
    Car(unsigned k=20) : Vehicle(k) { cout<<"hi\n"; }
    ~Car() { cout<<"chau\n"; }
    void drive() { km *= 2; cout<<km<<endl; }
};

class Truck : public Car{
    unsigned km;
public:
    Truck() : km(100) { cout<<"good morning\n"; }
    void drive(unsigned k) { km = km + 3*k; cout<<km<<endl; }
};

void main() {
    Vehicle* t = new Truck;
    Vehicle* c = new Car(10);
    t->drive(10);
    c->drive();
    delete t;
    delete c;
}
```

פלט:

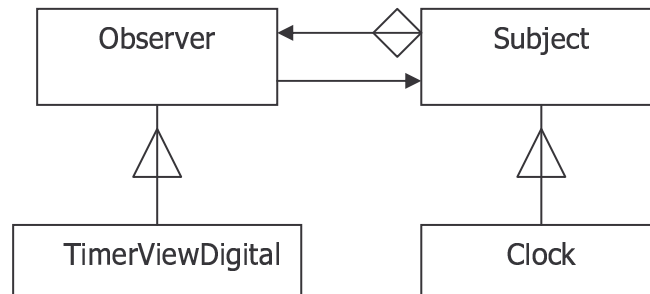
- hello** _____
- hi** _____
- good morning** _____
- hello** _____
- hi** _____
- 130** _____
- 11** _____
- bye bye** _____
- bye bye** _____
- _____
- _____
- _____



חלק ג'

שאלה 17: בחלק זה שאלה הפרושה על כמה סעיפים.

17. מוצע התיכון הבא:



- Observer הינו אובייקט המעוניין שיקראו לו כשמשהו מעניין קורא.
- Subject הינו אובייקט שמדווח ל- Observer שנרשם אליו לקבל עדכונים.
- ל- Subject יכולים להיות הרבה אובייקטי Observer שנרשמו אליו לקבל עדכונים.
- מהם יורשים

- Clock שהינו בעצם Subject, והוא דואג לדווח למי שנרשם אליו, כל שניה.
- TimerViewDigital שהינו בעצם Observer שמעוניין שיקראו לו כל שניה. הוא מתוכנן לבצע פעולה כלשהי כל X שניות, לכן כשנקרא על עידכון (כל שניה), הוא מבצע פילטור על השניות, וכשמגיע הזמן הרלוונטי מבצע את אשר צריך.
- אובייקטים
 - במערכת ישנו אובייקט גלובלי Clock (שהוא בעצם subject), שמתעורר כל שניה, ומדווח על כך לכל הרשומים.
 - אנחנו מייצרים 2 אובייקטים קונקרטיים TimerViewDigital, האחד אמור לדווח כל 2 שניות, והשני כל 3 שניות. 2 האובייקטים נרשמים אל ה- Clock ומקבלים שירותי "הערה" כל שניה.
- דוגמת הרצה: עבור ה- main במסגרת השמאלית יתקבל הפלט המופיע במסגרת הימנית:

main	output
<pre> #include "Observer.h" #include "Subject.h" int main() { //Clk1 is working with 2 secs resolution TimerViewDigital clk1("Clk1", 2); //Clk2 is working with 3 secs resolution TimerViewDigital clk2("Clk2", 3); TheClock.Go(); return 0; } </pre>	<pre> Clock at 1 ticks Clock at 2 ticks Clk1 has woken up! Clock at 3 ticks Clk2 has woken up! Clock at 4 ticks Clk1 has woken up! Clock at 5 ticks Clock at 6 ticks Clk1 has woken up! Clk2 has woken up! Clock at 7 ticks Clock at 8 ticks Clk1 has woken up! Clock at 9 ticks Clk2 has woken up! Clock at 10 ticks Clk1 has woken up! </pre>

Observer.h

```

#ifndef _OBSERVER_H_
#define _OBSERVER_H_

class Subject;

class Observer {
public:
    Observer(Subject& rSubject);
    virtual ~Observer() {}

    virtual void NotificationFromSubject(const Subject & rSubject) = 0;
    _____
    _____
    _____
    _____
    _____

};

class TimerViewDigital : public Observer {
    const char*   m_timerName;
    int           m_wakeUpTime;
public:
    _____
    _____
    _____
    _____
    _____
    _____

};

#endif // _OBSERVER_H_

```



Subject.h

```
#ifndef _SUBJECT_H_
#define _SUBJECT_H_
#include<windows.h>
#include <list>
using namespace std;

class Observer;

class Subject {
    list<Observer*>    m_observers;
public:
    Subject() {}
    virtual ~Subject () {}
    int    AddObserver(Observer* pObs);
    int    RemoveObserver(Observer* pObs);

    _____
    _____
    _____

protected:
    void    NotifyObservers();
};

class Clock : public Subject {
public:
    Clock() : m_numOfTicks(0)  { }
    virtual ~Clock() {}
    void Go() {
        while (1) {
            Sleep(1000); //sleep for 1 second (1000 msec) and then continue
            ++theClock;
        }
    }

    int WhatTime() const {    return m_numOfTicks;    }

    _____
    _____
    _____

private:
    int m_numOfTicks;
};

extern Clock theClock;

#endif // _SUBJECT_H_
```

- ז. להלן הדרישות מכל מודול
- i. Subject תומך בפונקציונליות הבאה
 1. הוסף observer – כך שיקבל נוטיפיקציות
 2. הסר observer – כך שיפסיק לקבל נוטיפיקציות
 3. יידע את כל ה- observers – שהשתנית, ומי שמעוניין שיעשה משהו בנידון
 - ii. Clock תומך בפונקציונליות הבאה
 1. התחל לעבוד – לספור כל שניה, ולדווח לרשומים
 2. מה השעה (כמה ticks עברו מתחילת הספירה)
 3. ... משהו שחסר וישאל בשאלות להלן
 - iii. Observer תומך בפונקציונליות הבאה
 1. עדכון הגיע
 - iv. TimerViewDigital תומך בפונקציונליות הבאה
 1. כשעדכון הגיע – בודק אם מס השניות שעברו תואם את הזמנים שעליהם אמור להגיב. אם כן מדפיס הדפסה (ראה פלט), אם לא מתעלם מההנוטיפיקציה.

17.1 השלימו את הקבצים Subject.h, Observer.h לפי הדרישות הנ"ל.

17.2 צרו את הקבצים Subject.cpp, Observer.cpp לפי הדרישות הנ"ל.



17.3 ענו על השאלות הבאות המתייחסות לקוד המוצג ולקוד שהושלם על ידכם. **שימו לב:**
חלק מהשאלות מבקשות השלמה נוספת של קוד.

17.3.1 בקובץ observer.h ההצהרה **class Subject;** - למה צריך אותה ומדוע היא מספיקה?

צריך אותה מאחר ומשתמשים בה במחלקה Observer. לא ניתן לעשות include לקובץ הכותר מאחר ואז תבצע הכללה הדדית (Observer ו- Subject צריכים להכיר זה את זה).

17.3.2 מתי מיוצר האובייקט הגלובלי **Clock theClock;** ? מי קורא לבנאי שלו?

מאחר וזהו אובייקט גלובלי, הוא נוצר עם תחילת הריצה של התוכנית. קריאה לבנאי שלו מתבצעת לפני הפקודה הראשונה ב- main

17.3.3 כשמייצרים את האובייקט הגלובלי **Clock theClock;** - מה סדר הקריאות והביצוע (מי נקרא ומה מתבצע) כתוצאה מכך?

Subject ואז Clock.

17.3.4 מה המשמעות של להגדיר מתודה וירטואלית כ- inline ?

אין משמעות לכך מאחר ומתודה inline נפרשת בזמן קומפילציה בעוד שמתודה וירטואלית מפורשת בזמן ריצה (ולכן לא יכולה להיות inline, למעט מקרים מיוחדים)

17.3.5 מה משמעות ההצהרה ומה ההשלכות שלה
virtual void NotificationFromSubject() = 0;

משמעות ההצהרה היא שזוהי מתודה וירטואלית טהורה ולכן המחלקה המגדירה אותה היא אבסטרקטית. ההשלכות – יש לממשה במחלקות היורשות כדי שניתן יהיה להגדיר אובייקטים.

17.3.6 מדוע אנו זקוקים ל- virtual Dtor ? מתי כן ומתי לא?

במצב של ירושה כדי להרוס את האובייקט המוצבע (ולא את אובייקט מחלקת הבסיס), עלינו להגדיר מפרק וירטואלי, בדומה למתודות וירטואליות אחרות.

17.3.7 הקוד ++theClock() במתודה Go() לא עובר קומפילציה . הוסף את האופרטור המתאים במקום הרלוונטי.

בקוד



Notification.cpp

```
#include "Observer.h"
#include "Subject.h"

int main()
{
    //Clk1 is working with 2 secs resolution
    TimerViewDigital o1("Clk1", 2);
    //Clk2 is working with 3 secs resolution
    TimerViewDigital o2("Clk2", 3);

    theClock.Go();

    return 0;
}
```

Observer.h

```
#ifndef _OBSERVER_H_
#define _OBSERVER_H_

class Subject;

class Observer {
public:
    Observer(Subject& rSubject);
    virtual ~Observer () {}

    virtual void NotificationFromSubject(const Subject & rSubject) = 0;
};

class TimerViewDigital : public Observer {
    const char* m_timerName;
    int m_wakeUpTime;
public:
    TimerViewDigital(const char* timerName, int wakeUpTime);
    virtual ~TimerViewDigital() {}

    virtual void NotificationFromSubject(const Subject & rSubject);
};

#endif // _OBSERVER_H_
```

Observer.cpp

```
#include "Observer.h"
#include "Subject.h"
```

```

#include <iostream>

Observer::Observer(Subject& rSubject)
{
    rSubject.AddObserver(this);
}

////////////////////////////////////
TimerViewDigital::TimerViewDigital(const char* timerName, int wakeUpTime) :
    m_timerName(timerName),
    m_wakeUpTime(wakeUpTime),
    Observer(theClock)
{
}

void TimerViewDigital::NotificationFromSubject(const Subject & rSubject)
{
    if (dynamic_cast<const Clock&>(rSubject).WhatTime() % m_wakeUpTime == 0)
        cout << " " << m_timerName << " has woken up!" << endl;
}

```

Subject.h

```

#ifndef _SUBJECT_H_
#define _SUBJECT_H_

#include <windows.h>
#include <vector>

using namespace std;

class Observer;

class Subject {
public:
    Subject();
    virtual ~Subject() {}
    int AddObserver(Observer* pObs);
    int RemoveObserver(Observer* pObs);
protected:
    void NotifyObservers();
private:
    vector<Observer*> m_observers;
};

class Clock : public Subject {
public:

```

```

Clock() : m_numOfTicks(0)
{}
virtual ~Clock() {}

int operator++ ();

void Go();

int WhatTime() const {
    return m_numOfTicks;
}

private:
    int m_numOfTicks;
};

extern Clock theClock;

#endif // _SUBJECT_H_

```

Subject.cpp

```

#include "Subject.h"
#include "Observer.h"
#include <iostream>

////////////////////////////////////
int Subject::AddObserver(Observer* pObs) {
    m_observers.push_back(pObs);
    return 0;
}

Subject::Subject()
{}

int Subject::RemoveObserver(Observer* pObs) {
    ...//
    return 0;
}

void Subject::NotifyObservers () {
    vector<Observer*>::const_iterator iter;
    for ( iter = m_observers.begin() ;
          iter != m_observers.end() ;
          iter++ )
    {
        (*iter)->NotificationFromSubject(*this);
    }
}
////////////////////////////////////

```



```

//Global object
Clock theClock;

int Clock::operator++ () {
    m_numOfTicks++;
    cout << "Clock at " << m_numOfTicks << " ticks" << endl;

    NotifyObservers();

    return m_numOfTicks;
}

void Clock::Go () {
    while (1)
    {
        //sleep for 1 second (1000 msec) and then continue
        Sleep(1000);
        ++theClock;
    }
}

```

בהצלחה

