סמסטר א' תשס"ד
מועד: א' 15/2/2004
משך הבחינה: 3½ שעות
חומר עזר: כל חומר עזר מותר

בחינה בקורס : **תכנות מכוון עצמים ושפת C++**

מרצים : **אמיר קירש, איריס רוזנבלום-גבר**

הנחיות כלליות לבחינה:

- המבחן מורכב משני חלקים:

  חלק א' כולל 7 שאלות אמריקאיות. משקל כל שאלה 7 נקודות, סה"כ: 49 נק'.

  חלק ב' כולל שאלת תכנות שמשקלה הכולל 51 נק'.

- חובה לתעד בשאלת התכנות כל פעולה לא ברורה שנעשית.
- בשאלות האמריקאיות יש לסמן תשובה אחת לכל שאלה בטבלה המצורפת. במידה ומספר תשובות נראות נכונות יש לסמן את התשובה הנכונה ביותר. אם נבחרה תשובה ו' (אף תשובה אינה נכונה), חובה לספק הסבר במקום המיועד לכך.
- בסיום המבחן יש לרשום מס' ת.ז. במקום המיועד לכך בטופס התשובות, לוודא שטופס התשובות נמצא יחד עם טופס המבחן ולא ניתק ממנו, ולהגישם בתוך מחברת הבחינה.
- המבחן הינו עם חומר פתוח. כל חומר עזר מותר למעט מחשבים ניידים. אין להעביר חומר עזר בין תלמידים במהלך המבחן.
- נא לכתוב בכתב קריא ולא מחובר.

בהצלחה !

טבלת תשובות לחלק האמריקאי

מס' ת.ז. : _____

חובה לספק הסבר עבור תשובה ו'

	א	ב	ג	ד	ה	ו
שאלה 1						
שאלה 2						
שאלה 3						
שאלה 4						
שאלה 5						
שאלה 6						
שאלה 7						

הסברים לתשובות

חובה לספק הסבר במידה ונבחרה תשובה ו' (אף תשובה אינה נכונה).
מותר לצרף הסבר גם עבור תשובות אחרות. אמנם **רק** תשובה נכונה תזכה בניקוד עבור כל שאלה, אולם ניתן יהיה להסתמך על ההסבר במסגרת ערעור, אם יידרש. מומלץ לצרף הסבר לתשובה אמריקאית במיוחד במקרים בהם נראה לך שתשובתך דורשת הסבר או נימוק.

שאלה 1 : _____

שאלה 2 : _____

שאלה 3 : _____

שאלה 4 : _____

שאלה 5 : _____

שאלה 6 : _____

שאלה 7 : _____

חלק א' – שאלות אמריקאיות (49 נק' – 7 נק' לכל שאלה)

שאלות 1-7 מתייחסות לקטע הקוד הבא:

```
1. class ReferenceCounter
2. {
3.    int* m_rc;
4. public:
5.    ReferenceCounter():m_rc(new int(1)){}
6.    ReferenceCounter(const ReferenceCounter& rc)
7.      {attach(rc);}
8.    virtual ~ReferenceCounter(){}
9.    virtual void free()=0;
10.     virtual const ReferenceCounter& operator=
11.       (const ReferenceCounter& rc)
12.     {
13.       if(this != &rc) {
14.           detach();
15.           attach(rc);
16.       }
17.       return *this;
18.     }
19.     virtual void attach(const ReferenceCounter& rc)
20.     {
21.       m_rc = rc.m_rc;
22.       ++(*m_rc);
23.     }
24.     virtual void detach()
25.     {
26.       if(--(*m_rc)==0) {
27.           delete m_rc;
28.           free();
29.       }
30.     }
31.   };
32.
33.   template <class G>
34.   class Array: public ReferenceCounter
35.   {
36.     template <class T>
37.     class innerArray
38.     {
39.       T* m_arr;
40.       int m_size;
41.     public:
```

```
42.         innerArray(int size):m_size(size)
43.         {
44.           m_arr = new T[m_size];
45.           cout<<"array: "<<(void*)m_arr<<endl;
46.         }
47.         void free()
48.         {
49.             cout<<"free: "<<(void*)m_arr<<endl;
50.             delete []m_arr;
51.         }
52.         friend class Array<T>;
53.       };
54.
55.       innerArray<G> m_innerArray;
56.
57.     public:
58.       explicit Array(int size):m_innerArray(size){}
59.       virtual ~Array(){detach();}
60.       void free(){m_innerArray.free();}
61.       G& operator[](int index)
62.       {
63.         return m_innerArray.m_arr[index];
64.       }
65.     };
66.
67.     void main()
68.     {
69.       Array<int> arr1(3);
70.       Array<int> arr2 = arr1;
71.       Array<int> arr3(5);
72.       arr3 = arr2;
73.       arr3[0] = 2;
74.       arr2[1] = arr3[0];
75.       arr1[2] = arr3[1] + arr2[0];
76.       for(int i=0; i<3; i++)
77.       {
78.         cout<<arr1[i]<<endl;
79.       }
80.     }
```

השאלות עצמן מתחילות בעמוד הבא.

שאלה 1

נניח ששורה מס' 70 הדפיסה: array: 0x002F0848

מה הודפס בשורה 69 ?

א. array: 0x002F0848

ב. array: <some other address>

ג. שורה 69 לא אמורה להדפיס דבר.

ד. שורה 70 לא אמורה להדפיס דבר, שורה 69 אמורה להדפיס – array: <some address>

ה. הן שורה 69 והן שורה 70 לא אמורות להדפיס דבר.

ו. אף אחת מהתשובות אינה נכונה.

שאלה 2

נניח שהיינו מחליפים את פרמטר ה-template בשורה 71 מ-int ל-float (הנחה זו תקפה לשאלה זו בלבד), כיצד הדבר היה משפיע על שורה 72?

א. השורה היתה נותנת שגיאת קומפילציה.

ב. השורה היתה נותנת שגיאת Linker.

ג. היה מופעל אופרטור השמה default-י במקום זה של מחלקת הבסיס ReferenceCounter.

ד. היה מופעל אופרטור השמה של מחלקת הבסיס ReferenceCounter בלבד.

ה. היה מופעל אופרטור השמה של המחלקת Array בלבד, ללא מעבר במחלקת הבסיס.

ו. אף תשובה אינה נכונה.

שאלה 3

שירן טוענת שניתן להחליף את השם התכנותי G של פרמטר ה-template במחלקה Array, ל-T, למרות שזהו שמו של פרמטר ה-template גם במחלקה הפנימית (לדבריה, הקומפיילר מספיק חכם וידע להסתדר).

מורן טוען ששורה 52 מיותרת מכיון שממילא ניתן לגשת ל-data members של המחלקה הפנימית innerArray מתוך המחלקה החיצונית Array.

נירן טוען שהמילה explicit בשורה 58 מיותרת מכיון שעל-מנת לחסום casting אוטומטי יש לציין explicit גם בשורה 42.

דירן טוען שהיה צריך להוסיף בשורה 58 את המילה virtual על-מנת לתמוך במערכים מעורבים (שיוכלו להכיל למשל גם int-ים וגם string-ים).

מי מהחבורה צודק?

א. שירן (היא גם מצטיינת דיקאן!).

ב. מורן ונירן (הם מפציצים בכל מבחן, חבל על הזמן).

ג. דירן (הוא חתיך כמו דוגמן).

ד. שירן ודירן (שהיו חברים פעם, מזמן, עד שהיא עשתה קעקוע בישבן).

ה. כולם צודקים.

ו. אף תשובה אינה נכונה.

שאלה 4

האם אפשר היה לרשום בשורה 63:

```
return m_innerArray[index];
```

א. לא – יגרום לתעופה בזמן ריצה.

ב. לא, מכיון שהערך המוחזר איננו תואם לחתימה של הפונקציה.

ג. כן, אבל ההדפסות ב-main היו משתנות.

ד. כן, וההדפסות ב-main היו נשארות בדיוק אותו דבר.

ה. כן, בתנאי שהיינו מממשים אופרטור [] מתאים במחלקה innerArray.

ו. אף תשובה אינה נכונה.

שאלה 5

כמה פעמים נעבור בשורה 27 במהלך הריצה של התוכנית ?

א. 0

ב. 1

ג. 2

ד. 3

ה. 4

ו. אף תשובה אינה נכונה.

שאלה 6

מורן שירן דירן ונירן התווכחו האם יכול להיווצר מצב שבו שינוי ערכים במערך אחד יגרום לשינוי גם במערך אחר.

מורן טען שזה לא יכול לקרות מכיון שמחלקת הבסיס ReferenceCounter דואגת להעתקה חכמה.

נירן מסכים עם מורן, אך טוען שהסיבה לכך שתרחיש זה לא יכול להתקיים נובע דווקא מהעובדה שהמחלקה innerArray מוגדרת ב-private של Array, ולכן לא ניתן להגיע אליה באופן ישיר.

שירן טוענת שמורן הוא קשקשן וכך גם נירן. כאשר שני מערכים מצביעים בעצם לאותה הקצאה (כתוצאה מהעתקה או השמה) ואז מכניסים ערך או משנים ערך באמצעות אופרטור [], יווצר מצב שהערך ישתנה בשני המערכים ולא רק במערך שאליו פנינו באמצעות האופרטור [].

דירן מסכים עם שירן. לטענתו מחלקת הבסיס ReferenceCounter מטפלת רק בשחרור זכרון חכם בעת הצורך (ע"י ספירת מספר האובייקטים המצביעים לאותה הקצאה ושחרור הזכרון רק כאשר מספר זה הינו 0), אך המחלקה ReferenceCounter איננה מטפלת בהפרדת ההצבעה של שני מערכים כאשר ניגשים לשנות אחד מהם.

מי מהחבורה צודק?

א. מורן.

ב. מורן ונירן.

ג. שירן.

ד. שירן ודירן.

ה. כולם טועים!

ו. אף תשובה אינה נכונה.

<u>שאלה 7</u>

מהו הפלט של הלולאה בשורות 76-79? (=> מסמל מעבר שורה).

א. 4 <= 2 <= 2

ב. ערך זבל <= 2 <= 2

ג. 0 <= 0 <= ערך זבל

ד. 0 <= ערך זבל <= ערך זבל

ה. ערך זבל <= ערך זבל <= ערך זבל

ו. אף אחת מהתשובות אינה נכונה.

חלק ב' – שאלת תכנות (51 נק')

מעוניינים לכתוב תוכנית שתדע לעבור על קובץ C ולהחליף קריאה ל-define-ים במימוש שלהם (בדיוק כמו שעושה ה-pre-compiler).

התוכנית תטפל כמובן גם ב-define של קבועים וגם ב-define של מאקרו-ים.

תזכורת:

```
#define SIZE 10                    <-- this is a const value
#define ADD(A, B)  (A)+(B)         <-- this is a macro

void main()
{
     cout<<SIZE;                   <-- uses the const value
     cout<<ADD(4, 5);              <-- uses the macro
}
```

לצורך כתיבת התוכנית חולקה העבודה לשני צוותים, צוות אחד יתעסק בריצה על הקובץ, זיהוי הגדרות ה-define וזיהוי הקריאות ל-define (להלן – צוות הקובץ), בעוד שהצוות השני יתעסק בהחלפת קריאה ל-define בתוכן שיש לשתול במקומו (להלן – צוות ה-define).

אתה נמנה על צוות ה-define ובאחריותך לממש באופן מדוייק את הממשק שאמור לשרת את צוות הקובץ.

להלן הקריאות שיבצע צוות הקובץ:

[1] בתחילת הריצה על קובץ:

```
DefineManager defineManager;
```

[2] כאשר תזוהה הגדרה של קבוע define:

```
defineManager.add(<define_name>, <define_substitute>);
// example:
defineManager.add("SIZE", "10");
```

[3] כאשר תזוהה הגדרה של מאקרו:

```
defineManager.add(<macro_name>, <param_list>, <macro>);
// example:
list<string> param_list;
param_list.insert(param_list.end(), string("A"));
param_list.insert(param_list.end(), string("B"));
defineManager.add("ADD", param_list, "(A)+(B)");
```

שתי הקריאות לעיל אמורות לזרוק Exception מתאים במידה ומנסים להוסיף הגדרת define (קבוע או מאקרו) עם שם תפוס (שנתפס כבר ע"י קבוע או ע"י מאקרו).

[4] כאשר תזוהה קריאה לקבוע define:

```
string substitute = defineManager.get(<define_name>);
// example:
string substitute = defineManager.get("SIZE");
// --> should return the string "10"
```

[5] כאשר תזוהה קריאה למאקרו:

```
string substitute =
     defineManager.get(<macro_name>, <param_list>);
// example:
list<string> param_list;
param_list.insert(param_list.end(), string("4"));
param_list.insert(param_list.end(), string("5"));
string substitute =
     defineManager.get("ADD", param_list);
// --> should return the string "(4)+(5)"
```

יש לשים לב שפרמטר מסויים יכול להיות בשימוש יותר מפעם אחת במאקרו ונדרש לטפל גם במצבים אלו. למשל:

```
#define SQR(x, y)  (x)*(x)+2*(x)*(y)+(y)*(y)

// the call for add will be (see [3] above):
list<string> param_list;
param_list.insert(param_list.end(), string("x"));
param_list.insert(param_list.end(), string("y"));
string macro = "(x)*(x)+2*(x)*(y)+(y)*(y)";
defineManager.add("SQR", param_list, macro);

// the call for getting a substitute for SQR(4, 5)
// will be (see [5] above):
list<string> param_list;
param_list.insert(param_list.end(), string("4"));
param_list.insert(param_list.end(), string("5"));
string substitute =
     defineManager.get("SQR", param_list);
// --> should return the string "(4)*(4)+2*(4)*(5)+(5)*(5)"
```

<u>שים לב</u>: סעיפי השאלה מתחילים בעמוד הבא (קרא בעיון את כל הסעיפים לפני התחלת הפתרון).

סעיף א' (27 נקודות)

כתוב את הגדרת המחלקות הנדרשות לפתרון השאלה ללא מימושים, פרט למימוש Constructors, Copy Constructor, Destructor ואופרטור השמה – במידה והינם נדרשים לדעתך.
(מותר לחסום את השימוש ב-Copy Constructor ובאופרטור השמה, במידה ולדעתך המימוש ה-default-י שלהם אינו טוב, והם אינם נדרשים לדעתך. במידה והם נדרשים – עליך לממשם).

הנחיות עזר:

- המחלקה DefineManager הינה כמובן הכרחית. בנוסף אליה נדרשות כנראה מחלקות נוספות לצורך החזקת הגדרות ה-define בתוך DefineManager.
- נראה הגיוני שהמחלקה DefineManager תשתמש במבנה נתונים STL-י מתאים.
- החלק היותר מסובך בשאלה הינו מימוש הפונקציה get עבור מאקרו-ים. יש לשים לב שמשקלה של פונקציה זו בשאלה הינו 6 נקודות בלבד (סעיף ה') ולכן רצוי לא להתמקד בנקודה זו, במיוחד לא בתחילת הפתרון.

סעיף ב' (6 נקודות)

כתוב את מימוש הפונקציה

```
void DefineManager::
add(const string& define_name, const string& define_substitute)
throw(DuplicateDefineNameException);
```

הפונקציה אמורה לאפשר הכנסת קבוע define חדש לתוך אובייקט DefineManager.
במידה ושם ה-define כבר תפוס (ע"י קבוע define אחר או ע"י מאקרו) הפונקציה תזרוק Exception מתאים (אין צורך לממש את מחלקת ה-Exception, ניתן להניח שהיא כבר קיימת).

סעיף ג' (6 נקודות)

כתוב את מימוש הפונקציה

```
void DefineManager::
add(const string& macro_name, const list<string>& param_list,
const string& macro) throw(DuplicateDefineNameException);
```

הפונקציה אמורה לאפשר הכנסת מאקרו חדש לתוך אובייקט DefineManager.
במידה ושם המאקרו כבר תפוס (ע"י קבוע define או ע"י מאקרו אחר) הפונקציה תזרוק Exception מתאים (אין צורך לממש את מחלקת ה-Exception, ניתן להניח שהיא כבר קיימת).

המשך בעמוד הבא

סעיף ד' (6 נקודות)

כתוב את מימוש הפונקציה

```
const string& DefineManager::
get(const string& define_name) throw(DefineNameNotFoundException);
```

הפונקציה אמורה להחזיר את ערך הקבוע (כאובייקט string) עבור define מסויים.
במידה ושם ה-define לא קיים הפונקציה תזרוק Exception מתאים (אין צורך לממש את מחלקת ה-Exception, ניתן להניח שהיא כבר קיימת).

סעיף ה' (6 נקודות)

כתוב את מימוש הפונקציה

```
string DefineManager::
get(const string& macro_name, const list<string>& param_list)
throw(DefineNameNotFoundException, WrongNumberOfArguments);
```

הפונקציה אמורה להחזיר ערך החלופי למאקרו כאובייקט string.
הפונקציה תזרוק Exception מתאים במקרים הבאים:

- במידה ושם המאקרו לא קיים.
- אם מספר הפרמטרים שנשלחו בפועל אינו תואם את מספר הפרמטרים שהוגדר במאקרו.

(אין צורך לממש את מחלקות ה-Exception, ניתן להניח שהן כבר קיימות).

<u>הנחיות עזר</u>:

- כחלק מהגדרות המאקרו שנמסרו בקריאה לפונקציה add, התקבלה גם רשימת הפרמטרים. למעשה מה שנדרש הוא לרוץ על רשימת הפרמטרים האמיתיים שנשלחו כעת ולהחליף את כל המופעים של הפרמטר הראשון בערך האמיתי שלו, וכך הלאה עבור הפרמטר השני וכו'.
- ניתן להניח שקיימת הפונקציה הבאה במחלקה string:

```
string string::substring(int startIndex, int endIndex);
```

הפונקציה מחזירה תת-מחרוזת פנימית מתוך אובייקט המחרוזת שקרא לפונקציה, החל מהאינדקס startIndex (כאשר 0 מציין את תחילת המחרוזת) ועד האינדקס endIndex-1.

- ניתן להניח שקיימת הפונקציה הבאה במחלקה string:

```
int string::indexOf(const string& string_to_look_for, int startIndex);
```

הפונקציה מחפשת הופעה של המחרוזת string_to_look_for בתוך אובייקט המחרוזת שקרא לפונקציה, החל מהמיקום startIndex (כאשר 0 מציין את תחילת המחרוזת).
במידה ומחרוזת החיפוש נמצאה מוחזר האינדקס שבו היא נמצאה.
במידה ומחרוזת החיפוש לא נמצאה מוחזר הערך 1-.

- יידרשו פונקציות עזר במחלקה מתאימה, למשל פונקציה בשם indexOfParameter שתשתמש בפונקציה indexOf של string אבל תדלג על הופעות של מחרוזות שהתו הבא אחריהן הוא אות או מספר (מה שאומר שהמחרוזת שנמצאה איננה הפרמטר שחיפשנו).

סוף !