אוניברסיטת בן-גוריון

מדור בחינות

מספר נבחן:_____

רשמו תשובותיכם בגיליון התשובות בלבד
תשובות מחוץ לגיליון לא יבדקו.
שימו לב:

על תשובות ריקות בשאלות הפתוחות יינתן 20% מהניקוד!

תאריך הבחינה: 26.2.2018
שם המורה: ד"ר מני אדלר
ד"ר ערן טרייסטר
מר בני לוטטי
פרופ' אנדרי שרף
שם הקורס: תכנות מערכות
מספר הקורס: 202-1-2031
מיועד לתלמידי: מדעי המחשב, הנדסת תוכנה
שנה: תשע"ח
סמסטר: א'
מועד: ב'
משך הבחינה: שלש שעות
חומר עזר: אסור

בהצלחה!

שאלה 1: ניהול זיכרון (30 נקודות)

(השאלה אינה מניחה שום דבר ממועד א'). בוגר SPL שהחליט לעבוד בחברת עיבוד תמונה נדרש לממש ב-C++ שתי מחלקות לתמונות שחור-לבן וצבע. כל ההבדל בינן הוא בטיפוס ערכי מערך הפיקסלים (שחור לבן - ערך יחיד לפיקסל, צבע - שלישיית ערכים). כמו כן, נדרש שיהיה טיפוס כללי (אבסטרקטי) עבור תמונה מסיבות ברורות. הוא קרא את המימוש ממועד א' ונחרד מכפילות הקוד.

הוא הציע את הקוד הבא:

```
class RGB {
        short value[3];
};

class AbstractImage{
protected:
        int n_pixels;
public:
        virtual void show() = 0;
        virtual AbstractImage* getMe() = 0;
        virtual ~AbstractImage(){};
};
```

```
template <typename T> class Image : public AbstractImage {
	T* data;
public:
	Image<T>(int n){n_pixels = n; data = new T[n_pixels]; }
	virtual ~Image<T>(){ delete[] data; }
	Image<T>(const Image<T>& rhs){
		n_pixels = rhs.n_pixels;
		data = new T[n_pixels];
		copyData(rhs);
	}
	Image<T>& operator=(const Image<T>& rhs) {
		n_pixels = rhs.n_pixels;
		delete[] data;
		data = new T[n_pixels];
		copyData(rhs);
		return *this;
	}
	virtual void show(){/* Some code here*/}
	virtual Image<T>* getMe(){return this;}
private:
	void copyData(const Image<T>& rhs){
		for (int i =0 ; i < n_pixels ; ++i){
			data[i] = rhs.data[i];
		}
	}
};
typedef class Image<RGB> ColorImage;
typedef class Image<short> BWImage;
```

<u>סעיף א'</u>

באופן כללי, אילו מהטענות הבאות הכי נכונה:

1. הרעיון הכללי של הקוד תקין, ואין שום בעיה עקרונית.
2. הרעיון הכללי של הקוד תקין, אבל הקומפיילר יחליף כל מופע של short עם מופע של המחלקה Short העוטפת את הטיפוס הפרימיטיבי (auto-boxing). ניתן לעשות זאת מראש ולקבל קוד מהיר יותר.
3. כעיקרון, כאשר משתמשים ב-typename אין בשום אופן צורך ב-virtual functions.
4. ה-destructor של Image<RGB>, Image<short> לא יכול להיות אותו הקוד, כיוון שיש לשחרר את אובייקטי ה-RGB אחד אחד בעזרת delete. בקוד הנוכחי יש זליגת זיכרון בטיפוס Image<RGB>.
5. אחרי הקומפילציה יהיה טיפוס יחיד Image<Object>, והקומפיילר יבצע המרות איפה שצריך בתכנית.

<u>סעיף ב'</u>

נתונה התכנית הבאה:

```
int main(){
        AbstractImage* A = new ColorImage(4);
        ColorImage B = ColorImage(4);
        A->show();
}
```

נתונה ההצעה למטה עבור מצב הזיכרון מיד לאחר הקריאה לפונקציה show() (תמונת הזיכרון נלקחת במקום שבו קיימת בקוד למעלה ההערה /* some code here */). סמנו את התשובה הנכונה ביותר.

מחסנית			ערימה		
כתובות	ערך	תיאור	כתובות	ערך	תיאור
100	5000	מצביע ל-A	5000	10000	A::vpointer (pointer to vtable)
104	10000	B::vpointer (to vtable)	5004	4	A::n_pixels
108	4	B::n_pixels	5008	8000	A::data
112	7000	B::data	7000	?????	תחילת מערך של 4 אובייקטי RGB.
116	20000	כתובת החזרה לmain	8000	?????	תחילת מערך של 4 אובייקטי RGB.
120	5000	מצביע ל-A (פרמטר this)			

1. הטבלאות לא נכונות כי לאובייקט A אין vtable ו-vpointer (מצביע ל-vtable).
2. הטבלאות לא נכונות כי בכתובות 112 צריך להיות כתוב "תחילת מערך של 4 אובייקטי RGB" ואילו כתובת 7000 אינה שייכת לתכנית בכלל. זאת כיוון שהאובייקט B הוקצה על המחסנית.
3. הטבלאות לא נכונות כי לפונקציה show אין פרמטרים. אין צורך להעביר את פרמטר this, וכתובת 120 אינה שייכת לתכנית.
4. הטבלאות לא נכונות כי שני שדות ה-1 אמורים להכיל כתובות שונות.
5. הטבלאות נכונות. בנוסף ישנן טבלאות וירטואליות בזיכרון, אחת מהן החל מכתובת 10000.

סעיף ג'

נסתכל על הקוד הבא:

```
ColorImage A = ColorImage(5); // (*)
ColorImage *B = &A;                    // (**)
A = *B;                                // (***)
```

אילו מהטענות הבאות נכונה:

1. בשורה (*) מופעל קוד של Copy constructor.
2. השורה (***) מפעילה קוד שגוי (שיש בו bug).
3. התכנית אינה יעילה. לא מומש move assignment operator ולכן יש העתקה של מערכים שלא לצורך.
4. התכנית אינה עובדת. לא מומש move assignment operator ולכן ישנה שגיאת קומפילציה בשורה (***).
5. טענות 1 ו-2 נכונות.

סעיף ד'

נדרש להוסיף למחלקה את הפונקציה getBrighterCopy(), אשר תחזיר העתק של האובייקט והמידע של התמונה בתוכו, אחרי הבהרה של התמונה בצורה כלשהי (שלא רלוונטית). מהנדס התלבט בין החתימות הבאות:

```
(i) Image<T> getBrighterCopy();
(ii) Image<T>& getBrighterCopy();
(iii) Image<T>* getBrighterCopy();
```

סמנו את התשובה הנכונה והאינפורמטיבית ביותר לדעתכם:

1. לגישה (i) יש יתרון ברור: האובייקט המוחזר יוקצה על המחסנית, וישוחרר באופן אוטומטי. אין צורך מהמשתמש לקרוא במיוחד ל-delete. מצד שני, חייבים בגישה זו לממש את חוק ה-5 כדי למנוע העתקות מיותרות.
2. בנוסף גישות (ii) ו-(iii) טובות, אך המשתמש חייב להבין שאם האובייקט הוקצה באופן דינאמי בתוך הפונקציה, אז האחריות לשחרר אותו (בעזרת delete) היא עליו (על המשתמש).
3. סעיף (2) נכון בהכרח רק עבור גישה (iii). משתנה המוחזר כרפרנס לא חייב להיות מוקצה דינאמית - תלוי במימוש. הוא יכול להיות מוקצה סטאטית בתוך הפונקציה, לחזור לפונקציה הקוראת, והקומפיילר ישחרר אותו בסוף הסקופ של הפונקציה הקוראת. זה ההבדל העקרוני בין מצביע (פוינטר) לרפרנס.
4. תשובות 1 ו-2 נכונות.
5. תשובות 1 ו-3 נכונות.

סעיף ה'

נדרש להוסיף למחלקה Image את המתודה הוירטואלית בעלת החתימה:

```
virtual void convertOrCopy(AbstractImage* rhs)
```

פונקציה זו תהיה הכללה של Copy Assignment Operator. במקרה וטיפוס של rhs* זהה לטיפוס האובייקט (this*), מבוצעת השמה "רגילה". במקרה ו-rhs* מטיפוס שונה משל האובייקט, מבוצעת המרה של טיפוסי התמונות (מצבע לשחור-לבן ולהיפך, כאשר בעתיד כנראה יוספו עוד טיפוסים כדוגמת Image<float>).
לשם הבהרה, הקוד הבא צריך לרוץ בצורה נכונה:

```
int main(){
        AbstractImage* A = new ColorImage(4);
        AbstractImage* B = new BWImage(4);
        AbstractImage* C = new ColorImage(4);
        B->convertOrCopy(A);                    // color image A is converted into BW image B
        A->convertOrCopy(B);                    // BW image B is converted into color image A
        C->convertOrCopy(A);                    // the same as *C = *A
        delete A; delete B; delete C;
}
```

לשם כך, הוספו הפונקציות הסטטיות הבאות למחלקה Image<T>:

```
void convert(Image<RGB>* lhs, Image<short>* rhs) {/* converting BW image to color*/}
void convert(Image<short>* lhs, Image<RGB>* rhs) {/* converting color image to BW*/}
void convert(Image<T>* lhs, Image<T>* rhs) {*lhs = *rhs;}
```

הפונקציות הללו מעתיקות/ממירות את rhs* אל תוך lhs*.

סמנו את התשובה הנכונה:

1. ניתן לממש את הפונקציה ConvertOrCopy בקלות ע"י המרות דינמיות כדוגמת הצורה הבאה:

```
if (dynamic_cast<Image<RGB>*>(rhs)){
        Image<T>::convert(this,dynamic_cast<Image<RGB>*> (rhs));
}
```

2. ניתן לממש את ConvertOrCopy בדומה לאפיון visitor pattern (או double dispatch) ע"י הוספת המתודות הבאות למחלקה Image<T>:

```
virtual void convertOrCopy(AbstractImage* rhs)  {
        rhs->convertOrCopyToRHS(this);
 }
virtual void convertOrCopyToRHS(Image<RGB>* rhs) {
         Image<T>::convert(rhs, this);
 }
virtual void convertOrCopyToRHS(Image<short>* rhs){
        Image<T>::convert(rhs, this);
}
```

כאשר יש להוסיף את המתודות האלו ללא מימוש (או כאבסטרקטיות) ל-AbstractImage.
שימו לב - הפונקציה getMe מופיעה במימוש בתחילת השאלה.

3. בעזרת dynamic_cast, הקומפיילר יודע לבצע את ההמרה לטיפוס הנכון מראש, כבר בזמן הקומפילציה.
4. תשובות 1 ו-3 נכונות.
5. תשובות 1 ו-2 נכונות.

שאלה 2: מקביליות (30 נקודות)

נתון הממשק **Stack**, המגדיר מחסנית בגודל חסום לשמירת אובייקטים (התומך בהכנסת ערכי null):

```
interface Stack<T> {

        //@INV: 0 <= size() <= capacity()

        void push(T data);    // the pushed data can be null
        T pop();
        int size();              // returns the current number of items in the stack
        int capacity();   // returns the maximal number of items which can be stored in the stack
}
```

נתון מימוש בטוח ונכון של הממשק, שאינו משתמש ב synchronized:

```
import java.util.concurrent.atomic.AtomicBoolean;

interface Stack<T> {
        void push(T data);
        T pop();
        int size();
        int capacity();
}

class Link<T> {

   public Link<T> next;
   public T data;

   public Link(Link<T> next, T data) {
      this.next = next;
      this.data = data;
   }
}
```

```
class LinkedStack<T> implements Stack<T> {

    private volatile Link<T> head;
    private volatile int size;
    private final int capacity;
    private AtomicBoolean isFree;

    LinkedStack(int capacity) throws Exception {
            if (capacity < 1)
                    throw new Exception("Illegal capacity: " + capacity);
            this.capacity = capacity;
            this.head = null;
            this.size = 0;
            this.isFree = new AtomicBoolean(true);
    }
    public void push(T data) {
            while (!isFree.compareAndSet(true,false));
            if (size < capacity) {
                    head = new Link<>(head, data);
                    size++;
            }
            isFree.set(true);
    }
    public T pop() {
            while (!isFree.compareAndSet(true,false));
            T ret = null;
            if (size > 0) {
                    ret = head.data;
                    head = head.next;
                    size--;
            }
            isFree.set(true);
            return ret;
    }
    public int size () {
            int ret;
            while (!isFree.compareAndSet(true,false));
            ret = size;
            isFree.set(true);
            return ret;
        }
```

```
    public int capacity () {
        return capacity;
    }

}
```

א. נוספה כעת דרישה כי ניתן יהיה להשתמש במחסנית בשני אופנים אפשריים:'גודל פתוח' - מספר איברים המחסנית אינו מוגבל, 'גודל חסום' - מספר האברים במחסנית מוגבל (כפי שזה כעת).

- הגדירו מחדש את התכונה הנשמרת (האינווריאנטה) כך שתתאים לדרישות החדשות. אם נדרש, יש להוסיף שאילתות לממשק
- הגדירו את תנאי ההתחלה והסיום של המתודות push, pop.

(10 נקודות)

ב. במימוש הנוכחי, אם לא מתקיימים תנאי ההתחלה של המתודות push/pop הפעולה לא מתבצעת.
עדכנו את הקוד כך שבמידה ותנאי ההתחלה אינם מתקיימים הת'רד עובר למצב blocked עד אשר הם יתקיימו, כדי לבצע t, הפעולה אח"כ.

- אין להשתמש ב synchronized, או במחלקות העושות שימוש ב synchronized (משמעות הדברים היא שלא ניתן להשתמש במנגנון wait/notify הדורשים סינכרון)
- אין להשתמש במחלקות ה- Condition, Lock, Semaphore של Java
- בניגוד למנגנון הסנכרון, הממומש במחלקה עם busy wait, <u>עבור מנגנון ההמתנה לקיום תנאי ההתחלה אין להשתמש ב busy wait</u> (כלומר, הת'רד צריך לעבור למצב blocked כאשר תנאי ההתחלה אינם מתקיימים, ולהתעורר רק כאשר יש אינדיקציה כלשהי לכך שתנאי ההתחלה עשויים להתקיים)

אין צורך להעתיק את הקוד הקיים - מספק לכתוב רק את התוספות והשינויים

<u>חומר עזר:</u> ניתן להניח כי קיימת במחלקה Thread מתודה אטומית sleepIfNotInterrupted המבצעת sleep לפרק זמן לא מוגבל אם ערך שדה ה interrupted הוא false [כלומר אם isInterrupted() == false], אחרת היא מבצעת השמה של false לשדה ה interrupted ולא מבצעת sleep.

[הסעיף מתייחס לקוד המקורי בשאלה - ניתן לענות על סעיף זה גם אם לא עניתם על סעיף א']

(15 נקודות)

ג. נתונים שני ת'רדים T1,T2 ושני אובייקטים S1,S2 מסוג Stack, הממומשים על פי המדיניות בסעיף ב' [ניתן לענות על סעיף זה, גם אם לא מימשתם את סעיף ב'].
הראו תרחיש בו שני הת'רדים מגיעים למצב של 'חֶבֶק' (deadlock) בעקבות קריאות למתודות של S1,S2.
(5 נקודות)

שאלה 3: תקשורת (30 נקודות)

בחברת ACME המשתמשת בReactor לצורך טיפול בהמון לקוחותיה נפגשו שרגא – ראש צוות הIT ונחום – ראש צוות הפיתוח לשיחת חולין בחדר האוכל.

א. שרגא סיפר לנחום שהוא השווה בין ביצועי הReactor לשרת ה TPC - Thread Per Client עליו למד בקורס SPL בצורה הבאה: ראשית הוא הריץ שרת TPC ושלח אליו ממחשבו האישי מספר בקשות ומדד את זמן התגובה שלו, אח"כ הוא הריץ שרת Reactor ושלח אליו את אותם בקשות בדיוק ומדד את זמן התגובה שלו. לתדהמתו הוא גילה כי זמן התגובה של TPC טוב יותר ולכן הסיק כי עדיף לחברה להחליף את ה Reactor בTPC. בתגובה הניף נחום את ידיו בזלזול ואמר כי המסקנה אליה הגיע שרגא שגוייה.
הסבירו, מדוע הגיב הReactor לאט יותר מהTPC במקרה של שרגא? בנוסף הסבר מה יכולות להיות הבעיות אם יחליטו בחברה להחליף את השרת שלהם מהReactor לTPC. (10 נקודות)

ב. בכל מחשב בחברה קיים שרת הודעות אליו המנכ"ל מוטי שולח משימות. נחום החליט להפנות את כל ההודעות שמגיעות לשרת שלו ישירות לשרגא ובכך לפנות חלק מזמנו (כי שרגא יחשוב שמוטי נתן לו את המשימות). בכדי לעשות כך הוא החליט לכתוב שרת הודעות משלו באמצעות הReactor. עזור לנחום לכתוב מחלקות הממשות MessageEncoderDecoder ו MessagingProtocol בשביל לבצע את מזימתו.
השלימו את הקוד שהספיק לכתוב נחום (הכתוב מטה) לפי ההנחות הבאות:

- פורמט ההודעה הוא שורה אחת של subject המכילה תווים באנגלית בלבד ונגמרת בתו '\n' ואז שורה אחת של body המכילה גם היא תווים באנגלית בלבד ונגמרת בתו '\n' (פורמט זה כמובן הוא אותו הפורמט בו תומכים כל שאר שרתי ההודעות בחברה ובפרט השרת של שרגא)
- הגודל הכולל של ההודעה אינו יכול להיות גדול יותר מ1000 תווים.
- שרת ההודעות של שרגא רץ על מחשב בעל כתובת 10.0.0.50(IP) ופורט 7777
- מפאת חוסר הבקיאות של שרגא בנושא הReactor החליט נחום שכל הודעה שכוללת את המילה reactor לא תופנה לשרגא אלא תודפס למסך.

<u>יש להשלים את הקוד בדף התשובות</u> (20 נקודות)

```
public class Mail {
  private String subject;
  private String body;
  public Mail(String subject, String body) {
        this.subject = subject;
        this.body = body;
  }
  public String getSubject() { return subject; }
  public String getBody() { return body; }
}
```

```
class MailEncoderDecoder implements MessageEncoderDecoder<Mail> {
  //Fields:
  ____________________
  ____________________
  ____________________

  public byte[] encode(Mail mail) {
        return (mail.getTitle() + "₩n" + mail.getBody() + "₩n").getBytes();
  }


  public Mail decodeNextByte(byte b) {
        ____________________________________________________
        ____________________________________________________
        ____________________________________________________
        ____________________________________________________
        ____________________________________________________
        ____________________________________________________
        ____________________________________________________
  }
}

 class MailMessagingProtocol implements MessagingProtocol<Mail> {
        public boolean shouldTerminate() { return false; }
        public Mail process(Mail mail) throws IOException {
          if (mail.getTitle().contains("reactor") || mail.getBody().contains("reactor")) {
            System.out.println("Message: {title=" + mail.getTitle() + ", body=" + mail.getBody() + "}");
          } else {
            Socket socketToShraga = ____________________________________
            OutputStream outToShraga = ________________________________
             ____________________________________________________
            ____________________________________________________
            ____________________________________________________
            socketToShraga.close();
          }
          ____________________________________________________________
          ____________________________________________________________
        }
}
```

שאלה 4: בסיסי נתונים (10 נקודות)

כחלק ממהפכת המחשוב שעוברת המשטרה, הוחלט לארגן את כל נתוני יחידת ההאזנה במסד נתונים רלציוני. האזנה מסוימת מאופיינת בהקלטה (נשמרת כטקסט), מספר טלפון ממנו נעשתה השיחה ומספר טלפון לאן שחויג ותאריך. כמו כן שמורים לכל מספר טלפון נתוני הבעלים שלו.

```
CREATE TABLE phone (
phone_num        INT        PRIMARY KEY,
name             TEXT       NOT NULL,
address          TEXT       NOT NULL
);

CREATE TABLE conversations (
phone_from       INT    NOT NULL,
phone_to         INT    NOT NULL,
text             TEXT   NOT NULL,
FOREIGN KEY(phone_from)  REFERENCES phone (phone_num),
FOREIGN KEY(phone_to)  REFERENCES phone (phone_num)
);
```

א. רוצים לאתר את הקלטות כל השיחות 'הכפולות' אשר חויגו מנתניה. שיחה 'כפולה' מוגדרת ע"י המשטרה כשיחה ממספר א' למספר ב' שעבורה קיימת שיחה ממספר ב' למספר א'.

<u>שימו לב</u>: המילה AS מאפשרת לבצע aliasing בSQL ולתת לטבלה שם זמני לצורך השאילתה באופן הבא:

```
SELECT  FROM table-name AS alias-name
```

בחרו את השאילתה המחזירה שיחות אלו:

1. SELECT conA.text, conB.text FROM conversations AS conA JOIN phone ON phone_num = conA.phone_from JOIN conversations AS conB ON conA.phone_to = conB.phone_from AND conB.phone_to = conA.phone_from AND phone.address = 'netanyah';

2. SELECT conA.text, conB.text FROM conversations AS conA JOIN phone ON phone_num = conA.phone_from JOIN conversations AS conB ON conA.phone_to = conB.phone_from WHERE phone.address = 'netanyah';

3. SELECT conA.text, conB.text FROM conversations AS conB JOIN phone ON phone_num = conB.phone_from JOIN conversations AS conA ON conA.phone_to = conB.phone_from AND phone.address = 'netanyah';

4. SELECT conA.text, conB.text FROM conversations AS conA JOIN conversations AS conB ON conA.phone_to = conB.phone_from AND conB.phone_to = conA.phone_from AND phone.address = 'netanyah';

5. לא ניתן לבצע את השאילתא הנדרשת

ב. השוטר אזולאי הציע להכליל את מושג השיחה 'הכפולה' לשיחה 'מעגלית'. להגדרתו, זוהי שיחה היוצאת ממספר א' למספר ב' שעבורה ישנה שיחה ממספר ב' למספר ג' וכך הלאה, עד אשר קיימת שיחה כלשהי החוזרת למספר א'. בחר את השאילתה המחזירה שיחות אלו

1. SELECT conA.text FROM conversations AS conA JOIN conversations AS conB JOIN conversations AS conC ON conA.phone_to = conB.phone_from AND conB.phone_to = conC.phone_from AND conC.phone_to = conA.phone_from;
2. SELECT conA.text, conB.text, conC.text FROM conversations AS conA JOIN conversations AS conB JOIN conversations AS conC ON conA.phone_from = conB.phone_to AND conB.phone_from = conC.phone_to AND conC.phone_from = conA.phone_to;
3. SELECT conA.text, conC.text, conB.text FROM conversations AS conA JOIN conversations AS conC JOIN conversations AS conB ON conA.phone_to = conB.phone_from AND conB.phone_to = conC.phone_from AND conC.phone_to = conA.phone_from;
4. תשובות 1,3 נכונות
5. לא ניתן לבצע את השאילתא הנדרשת