אוניברסיטת בן-גוריון

מדור בחינות

מספר נבחן:_____

רשמו תשובותיכם בגיליון התשובות בלבד
תשובות מחוץ לגיליון לא יבדקו.

בהצלחה!

תאריך הבחינה: 18.02.2019
שם המורה: פרופ' אנדרי שרף
ד"ר מני אדלר
ד"ר ערן טרייסטר
מר מג'יד קסיס
שם הקורס: תכנות מערכות
מספר הקורס: 202-1-2031
מיועד לתלמידי: מדעי המחשב, הנדסת תוכנה
שנה: תשע"ט
סמסטר: א
מועד: ב
משך הבחינה: שלוש שעות
חומר עזר: אין

שאלה 1 (30 נקודות)

רוצים לכתוב תוכנית c++ המאפשרת בדיקה של אפליקציות שונות (מסוגים שונים) יחד. נתון המימוש החלקי הבא:

```
class TESTER{
    virtual void test(APP *app) = 0;
    virtual void test(PAY_APP *app) = 0;
    virtual void test(PLAY_APP *app) = 0;
};
class APP{
    virtual void testMe(TESTER *tester) = 0;
};
class PAY_APP : public APP{
    virtual void testMe(TESTER *tester) { ???}
};
class PLAY_APP : public APP{
    virtual void testMe(TESTER *tester) { ???}
};
class PERFORMANCE_TESTER: public TESTER{
    virtual void test(APP *app) {???}
    virtual void test(PAY_APP *app) {/*pay app performance test*/ ; }
    virtual void test(PLAY_APP *app) { /*play app performance test*/; }
};
class FUN_TESTER: public TESTER{
    virtual void test(APP *app) {???}
```

```
    virtual void test(PAY_APP *app) {/*pay app fun test*/ ; }
    virtual void test(PLAY_APP *app) { /*play app fun test*/; }
};

class MY_APPS_REPO{
    MY_APPS_REPO(int n_play_apps, int n_pay_apps){
       apps_repo_arr_ = new APP*[n_play_apps+n_pay_apps];
       for (int i = 0; i < n_play_apps; i++)
            apps_repo_arr_[i] = new PLAY_APP();
       for (int i = n_play_apps; i < n_play_apps+n_pay_apps; i++)
            apps_repo_arr_[i] = new PAY_APP();
    }
    void test_apps_repo(MY_APPS_REPO& repo, TESTER *tester);

    APP **apps_repo_arr_;
    int apps_repo_size_;
};
```

א. _apps_repo_arr הוא מערך המכיל פוינטרים לאובייקטים מסוג PAY_APP ו-PLAY_APP אשר מוקצים ומאותחלים תקנית בבנאי. השלימו את הקוד החסר (במקומות המצויינים ב??? בלבד) על מנת שיהיה ניתן לבדוק את האובייקטים המאוחסנים על ידי קריאה לפונקציית ()test המתאימה לכל אובייקט בקוד הכתוב מטה. [15 נק.]

```
void MY_APPS_REPO::test_apps_repo(MY_APPS_REPO &repo, TESTER *tester){
    int i;
    //here
    for (i = 0; i < apps_repo_size_; i++) {
            tester->test(apps_repo_arr_[i]);
        }
    }
```

ב. נתון קטע הקוד הבא:

```
void main(){
      MY_APPS_REPO *repo = new MY_APPS_REPO(2,3);
      PERFORMANCE_TESTER *tester = new PERFORMANCE_TESTER();
      repo->test_apps_repo(*repo, tester);
}
```

ציירו את תמונת הזיכרון (STACK, HEAP) עבור הרצת הקוד עד //here בקוד? [10 נק.]

ג. ממשו בנאי הורס עבור MY_APPS_REPO המבצע מחיקה עמוקה. כיצד ניתן להבטיח כי נקרא הבנאי הורס המתאים עבור כל אובייקט בתוך ה- _apps_repo_arr?
[5 נק.]

שאלה 2 (30 נקודות)

נתון הממשק SortedBinaryTree, המייצג עץ בינארי שערכיו ממויינים ('עץ חיפוש בינארי'):

```
interface SortedBinaryTree<T extends Comparable<T>> {

   //@INV: getData() != null &&
   (getLeft() == null || getData().compareTo(getLeft().getData()) > 0) &&
   (getRight() == null || getData().compareTo(getRight().getData()) < 0)

      T getData();
      SortedBinaryTree<T> getLeft();
      SortedBinaryTree<T> getRight();
      void insert(T data) throws Exception;
      void remove(T data) throws Exception;
}
```

המתודה insert מקבלת קודקוד ומוסיפה אותו לעץ במקום שתואם את הקריטריון של עץ ממויין, כפי שהוגדר באינווריאנטה.
המתודה remove מקבלת ערך ומסירה את תת העץ מתחת לקודקוד בעל הערך הזה.
לדוגמא, הפעלת remove 5 על העץ הבא, תסיר את הקודקודים מתחת לקודקוד עם הערך 5:

א. הגדירו את תנאי ההתחלה והסיום של המתודות insert, remove (אם נדרש, הרחיבו את הממשק בהתאם (אין צורך לממש)), וציינו האם הם בהכרח מתקיימים בהרצה מקבילית, עבור המימוש הבא:

```
class SortedBinaryTreeImpl<T extends Comparable<T>>
                              implements SortedBinaryTree<T>, Iterable<T> {

      private volatile T data;
      private SortedBinaryTree<T> left, right;

      SortedBinaryTreeImpl(T data) throws Exception {
            if (data == null)
                  throw new Exception("Null value!");
            this.data = data; this.left = null; this.right = null; }

      public synchronized T getData() { return data; }
      public synchronized SortedBinaryTree<T> getLeft() { return left; }
      public synchronized SortedBinaryTree<T> getRight() { return right; }

      public synchronized void remove(T data) throws Exception {
            if (data == null)
                  throw new Exception("Null value!");

            if (this.data.compareTo(data) == 0) {
                  left = null;
                  right = null;
            } else {
                  if (this.data.compareTo(data) < 0)
                        if (right != null) right.remove(data);
                  if (this.data.compareTo(data) > 0)
                        if (left != null) left.remove(data);
            }
      }

      public synchronized void insert(T data) throws Exception {
            if (data == null)
                  throw new Exception("Null value!");

            if (this.data.compareTo(data) > 0) {
                  if (left == null)
                        left = new SortedBinaryTreeImpl<T>(data);
                  else
                        left.insert(data);
            }
            if (this.data.compareTo(data) < 0) {
                  if (right == null)
```

```
                        right = new SortedBinaryTreeImpl<T>(data);
                    else
                        right.insert(data);
                }
            }
    }
```

[10 נקודות]

נתונה המחלקה SortedBinaryIntTree, המרחיבה את המחלקה SortedBinaryTreeImpl עבור מספרים שלמים עם מתודת sum, וממשת את הממשק Iterable:

```
class SortedBinaryIntTree extends SortedBinaryTreeImpl<Integer>
                          implements Iterable<Integer> {

    SortedBinaryIntTree(Integer data) throws Exception {
       super(data);
    }

    public int sum() {
        int ret =0;
        Iterator<Integer> it = iterator();
        while (it.hasNext())
           ret += it.next();
        return ret;
     }

    public Iterator<Integer> iterator() {
        return new Iterator<Integer> () {

            SortedBinaryTree<Integer> currNode = SortedBinaryIntTree.this;
            Stack<SortedBinaryTree<Integer>> stack =
                                    new Stack<SortedBinaryTree<Integer>>();

            public Integer next() {
                while (currNode != null) {
                    stack.push(currNode);
                    currNode = currNode.getLeft();
                }
                currNode = stack.pop();
                Integer ret = currNode.getData();
```

```
                currNode = currNode.getRight();
                return ret;
            }

            public boolean hasNext() {
                return (!stack.isEmpty() || currNode != null);
            }
        };
    }
}
```

ב. הגדירו תנאי התחלה וסיום עבור המתודה sum (אם נדרש, הרחיבו את הממשק בהתאם (אין צורך לממש)), וציינו תרחיש מקבילי בו תנאי הסיום אינם מתקיימים.
הערה: אין צורך להתעמק במימוש מתודת ה next של האיטרטור, מלבד העובדה שעבור כל קודקוד נאגרים בזה אחר זה קודקודים המופעים מתחתיו.
[10 נקודות]

ג. עדכנו את המחלקה SortedBinaryIntTree, כך שתנאי הסיום של sum יתקיימו תמיד.
<u>אין לנעול אובייקט לכל זמן ביצוע המתודה sum או לכל זמן העתקת העץ</u>.
[10 נקודות]

שאלה 3 (25 נקודות)

בנספח למבחן מופיעה תבנית קוד הריאקטור, כפי שנלמדה בכיתה. בכל סעיף יש לתת הסבר **מלא**. הסבר **כללי** יקבל ציון חלקי. הסבר מלא יכיל את כל המידע הרלוונטי לסעיף לרבות שמות פונקציות, שורות קוד עיקריות, ומבני נתונים חיוניים. בשאלות של קוד אפשר לשנות ולהוסיף כל פונקציה שתרצו כל עוד ששומרים על חלוקה נכונה של משימות בשרת ונכונותו.

א. (1) **שנו את המימוש** של הפונקציות **`execute`** **ו-`complete`** (לפי צורך) כך שאם הפונקציה תסיים לטפל ב-Request כלשהו, היא תתחיל לטפל ב-Request הבא **ללא יצירה של Runnable חדש,** במקום זאת להריץ אותו בצורה סדרתית. הפונקציה תעשה זאת כל עוד יש Requests הממתינים לטיפול עבור לקוח זה, אחרת תצא בצורה תקינה. **[6 נקודות]**

(2) לאיזה עקרון מנוגד המימוש המוצע בסעיף (1)? תנו דוגמה כדי להמחיש את חומרת הבעיה. **[2 נקודות]**

ב. (1) הסבירו את תהליך הטיפול ב-`Event` מסוג `Write` מרגע קבלתו עד סיום הטיפול בו. **[5 נקודות]**

(2) מהו המספר **המינימלי** של `Responses` שיטופלו בעת קבלת Event **אחד** מסוג `Write`? מהו המספר **המקסימלי** של `Responses` שיטופלו במקרה כזה? תנו דוגמא עבור כל אחד משני המקרים. **[4 נקודות]**

ג. בפונקציה `pendingRunnablesOf` של `ActorThreadPool`, אנו משתמשים ב-`ReentrantReadWriteLock`.

(1) הסבירו סוג מנעול זה, הדגישו את ההבדלים בין השימוש במנעול ה-`Read` שלו, והשימוש במנעול ה-`Write` שלו, מה ההשפעה של נעילת כל סוג מהם? הדגימו כל אחד מהמקרים. **[4 נקודות]**

(2) הסבירו את הסיבה בגינה השתמשנו במנעול זה כדי לסנכרן את שורת הקוד המפעילה את הפונקציה `get` של ה-`WeakHashMap` של `actors`. תנו דוגמה הממחישה את הכשל הפוטנציאלי בשרת אם **לא** היינו משתמשים בנעילה זו. **[4 נקודות]**

שאלה 4 (15 נקודות)

נציגי חברת AirBNB פנו אליכם בבקשה ל persistence layer עבורם. צריך לתמוך במידע הבא:

1. יש משתמשים (users) אשר חלקם רק שוכרים דירות/חדרים (properties), חלקם מארחים (hosts), וחלקם שניהם. לכל משתמש יש מספר סידורי id.
2. לכל property יש מספר סידורי property_id, מארח (host) יחיד, מחיר ללילה ותיאור. משתמש יכול להיות מארח (host) של מספר properties.
3. השכרה (rental) מיוצגת ע"י אורח (guest), דירה/חדר (property) תאריך כניסה ויציאה. לא ניתן במערכת לחלוק השכרה בדירה/חדר ע"י כמה אורחים ללילה נתון.
4. בסיום כל השכרה (תאריך check_out), נכתבות המלצות. המארח ממליץ על האורח, והאורח ממליץ על המארח בהקשר לדירה הרלוונטית (בעצם זו המלצה על הproperty).

סעיף א [3 נקודות]:

בחברה השתמשו בשאילתות הבאות ליצירת הטבלאות. מצאו דוגמא לקשר בין שדות המייצג יחס של one-to-many. דוגמא שראינו בכיתה: "מספר סטודנטים רשומים למחלקה אחת".

```
CREATE TABLE Users (
   id            INT       PRIMARY KEY,
   name          TEXT      NOT NULL
);
CREATE TABLE Properties (
   property_id        INT     PRIMARY KEY,
   host_id            INT     NOT NULL,
   description        TEXT    NOT NULL,
   price_per_night    INT     NOT NULL,
   FOREIGN KEY(host_id)    REFERENCES Users(id)
);
CREATE TABLE Rentals (
   guest_id           INT     NOT NULL,
   property_id        INT     NOT NULL,
   check_in           DATE    NOT NULL,
   check_out          DATE    NOT NULL,
   rec_on_host        TEXT,
   rec_on_guest       TEXT,
   FOREIGN KEY(guest_id)      REFERENCES Users(id)
   FOREIGN KEY(property_id) REFERENCES Properties(property_id)
   PRIMARY KEY(property_id,check_in)
);
```

סעיף ב' [6 נקודות]:

רוצים כעת שאילתא לעדכון המלצה (recommendation) על אורח בטבלת ה-Rentals (עדכון השדה rec_on_guest). ראו דוגמא לשימוש בפקודה UPDATE למטה.
השלימו את קוד הפונקציה update_rec_on_guest() במחלקת ה-DAO הבאה:

```
class _Rentals:
     def update_rec_on_guest
               (self, guest_id, property_id, check_out, rec):
     c = self._conn.cursor()
     c.execute("""
     _______________FILL IN ANSWER SHEET_______________
     _______________FILL IN ANSWER SHEET_______________
     _______________FILL IN ANSWER SHEET_______________

     """, [________FILL IN ANSWER SHEET_________________])
```

לנוחיותכם, הקוד הבא הוא דוגמא למתודת DAO עם עדכון שם משתמש בטבלה Users

```
class _Users (object)
     def update_name(self,id,name):
        self._conn.cursor().execute("""
           UPDATE Users SET name = (?) WHERE id= (?)
                """,[name,id])
```

סעיף ג [6 נקודות]:

להנהלת חשבונות רוצים כעת שאילתא שבהינתן host_id מסוים, ושנה מסוימת year, תייבא את כל ההשכרות של דירות בבעלות המארח בעל ה-host_id, שתאריך ה-check_out שלהן באותה השנה.
יש לייבא את property_id, תאריך ה-check_out, תאריך ה-check_in, ואת המחיר ללילה (price_per_night) של כל השכרה רלוונטית. רשמו את השאילתא (אין צורך בקוד פייתון).

להזכירכם, ניתן להתייחס למשתנה year בSQL ע"י $year. כמו כן, בדוגמא הבאה "שולפים" שנה מתוך תאריך ומשווים את הערך שלה:

```
SELECT * FROM mytable WHERE YEAR(transaction_day)=2013
```

<u>נספח</u>: קוד הריאקטור, כפי שנלמד בכיתה

```
public class NonBlockingConnectionHandler<T> implements
                                            ConnectionHandler<T> {
  private static final int BUFFER_ALLOCATION_SIZE = 1 << 13; //8k
  private static final ConcurrentLinkedQueue<ByteBuffer> BUFFER_POOL =
                                        new ConcurrentLinkedQueue<>();

  private final MessagingProtocol<T> protocol;
  private final MessageEncoderDecoder<T> encdec;
  private final Queue<ByteBuffer> writeQueue = new
                                            ConcurrentLinkedQueue<>();
  private final SocketChannel chan;
  private final Reactor reactor;

  public NonBlockingConnectionHandler(
      MessageEncoderDecoder<T> reader,
      MessagingProtocol<T> protocol,
      SocketChannel chan,
      Reactor reactor) {
        this.chan = chan;
        this.encdec = reader;
        this.protocol = protocol;
        this.reactor = reactor;
  }

  public Runnable continueRead() {
    ByteBuffer buf = leaseBuffer();
    boolean success = false;
    try {
      success = chan.read(buf) != -1;
    } catch (IOException ex) {
      ex.printStackTrace();
    }
    if (success) {
      buf.flip();
      return () -> {
        try {
          while (buf.hasRemaining()) {
            T nextMessage = encdec.decodeNextByte(buf.get());
            if (nextMessage != null) {
              T response = protocol.process(nextMessage);
```

```
            if (response != null) {
              writeQueue.add(ByteBuffer.wrap(
                                encdec.encode(response)));
              reactor.updateInterestedOps(chan, SelectionKey.OP_READ
                                          | SelectionKey.OP_WRITE);
              }
            }
          }
        } finally {
          releaseBuffer(buf);
        }
      };
    } else {
      releaseBuffer(buf);
      close();
      return null;
    }
  }

public void close() {
  try {
    chan.close();
  } catch (IOException ex) {
      ex.printStackTrace();
  }
}

public void continueWrite() {
  while (!writeQueue.isEmpty()) {
    try {
      ByteBuffer top = writeQueue.peek();
      chan.write(top);
      if (top.hasRemaining()) {
        return;
      } else {
        writeQueue.remove();
      }
    } catch (IOException ex) {
      ex.printStackTrace();
      close();
    }
  }
```

```
    if (writeQueue.isEmpty()) {
      if (protocol.shouldTerminate()) close();
        else reactor.updateInterestedOps(chan, SelectionKey.OP_READ);
    }
  }

  private static ByteBuffer leaseBuffer() {
    ByteBuffer buff = BUFFER_POOL.poll();
    if (buff == null) {
      return ByteBuffer.allocateDirect(BUFFER_ALLOCATION_SIZE);
    }
    buff.clear();
    return buff;
  }

  private static void releaseBuffer(ByteBuffer buff) {
    BUFFER_POOL.add(buff);
  }
}
```

```java
public class ActorThreadPool<T> {

  private final Map<T, Queue<Runnable>> actors;
  private final ReadWriteLock actorsReadWriteLock;
  private final Set<T> playingNow;
  private final ExecutorService threads;

  public ActorThreadPool(int threads) {
      this.threads = Executors.newFixedThreadPool(threads);
      actors = new WeakHashMap<>();
      playingNow = ConcurrentHashMap.newKeySet();
      actorsReadWriteLock = new ReentrantReadWriteLock();
  }

  public void shutdown() {
    threads.shutdownNow();
  }

  public void submit(T actor, Runnable r) {
    synchronized (actor) {
      if (!playingNow.contains(actor)) {
        playingNow.add(actor);
        execute(r, actor);
      } else {
        pendingRunnablesOf(actor).add(r);
      }
    }
  }

  private Queue<Runnable> pendingRunnablesOf(T actors) {
    actorsReadWriteLock.readLock().lock();
    Queue<Runnable> pendingRunnables = actors.get(actors);
    actorsReadWriteLock.readLock().unlock();

    if (pendingRunnables == null) {
      actorsReadWriteLock.writeLock().lock();
      actors.put(actor, pendingRunnables = new LinkedList<>());
      actorsReadWriteLock.writeLock().unlock();
    }
    return pendingRunnables;
  }
```

```
  private void execute(Runnable r, T actor) {
    threads.submit(() -> {
      try {
        r.run();
      } finally {
        complete(actor);
      }
    });
  }

  private void complete(T actor) {
    synchronized (actor) {
      Queue<Runnable> pending = pendingRunnablesOf(actor);
      if (pending.isEmpty()) {
        playingNow.remove(actor);
      } else {
        execute(pending.poll(), actor);
      }
    }
  }
}
```

```java
public class Reactor<T> implements Server<T> {

    private final int port;
    private final Supplier<MessagingProtocol<T>> protocolFactory;
    private final Supplier<MessageEncoderDecoder<T>> readerFactory;
    private final ActorThreadPool<NonBlockingConnectionHandler> pool;
    private Selector selector;

    private Thread selectorThread;
    private final ConcurrentLinkedQueue<Runnable> selectorTasks = new
                                            ConcurrentLinkedQueue<>();

    public Reactor(
            int numThreads,
            int port,
            Supplier<MessagingProtocol<T>> protocolFactory,
            Supplier<MessageEncoderDecoder<T>> readerFactory) {

        this.pool = new ActorThreadPool<>(numThreads);
        this.port = port;
        this.protocolFactory = protocolFactory;
        this.readerFactory = readerFactory;
    }

    public void serve() {
        selectorThread = Thread.currentThread();

        try (   Selector selector = Selector.open();
                ServerSocketChannel serverSock =
                                    ServerSocketChannel.open()) {

            this.selector = selector; //just to be able to close

            serverSock.bind(new InetSocketAddress(port));
            serverSock.configureBlocking(false);
            serverSock.register(selector, SelectionKey.OP_ACCEPT);

            while (!Thread.currentThread().isInterrupted()) {

                selector.select();
                runSelectionThreadTasks();
```

```
                for (SelectionKey key : selector.selectedKeys()) {

                    if (!key.isValid()) {
                        continue;
                    } else if (key.isAcceptable()) {
                        handleAccept(serverSock, selector);
                    } else {
                        handleReadWrite(key);
                    }
                }

                selector.selectedKeys().clear();

            }

        } catch (ClosedSelectorException ex) {
            //do nothing - server was requested to be closed
        } catch (IOException ex) {
            //this is an error
            ex.printStackTrace();
        }

        System.out.println("server closed!!!");
        pool.shutdown();
    }

    void updateInterestedOps(SocketChannel chan, int ops) {
        final SelectionKey key = chan.keyFor(selector);
        if (Thread.currentThread() == selectorThread) {
            key.interestOps(ops);
        } else {
            selectorTasks.add(() -> {
                if(key.isValid())
                    key.interestOps(ops);
            });
            selector.wakeup();
        }
    }



private void handleAccept(ServerSocketChannel serverChan,
```

```
                        Selector selector) throws IOException {
        SocketChannel clientChan = serverChan.accept();

        clientChan.configureBlocking(false);
        final NonBlockingConnectionHandler<T> handler = new
             NonBlockingConnectionHandler<>(
                readerFactory.get(),
                protocolFactory.get(),
                clientChan,
                this);

        clientChan.register(selector, SelectionKey.OP_READ, handler);
    }

 private void handleReadWrite(SelectionKey key) {
        @SuppressWarnings("unchecked")
        NonBlockingConnectionHandler<T> handler =
                    (NonBlockingConnectionHandler<T>) key.attachment();
        if (key.isReadable()) {
            Runnable task = handler.continueRead();
            if (task != null) {
                pool.submit(handler, task);
            } else if (!key.isValid()) {
                return;
            }
        }
        if (key.isWritable()) {
            handler.continueWrite();
        }
    }

  private void runSelectionThreadTasks() {
    while (!selectorTasks.isEmpty()) {
      selectorTasks.remove().run();
    }
  }

  public void close() throws IOException {
    selector.close();
  }
}
```