**דגשים וטיפים לקראת הבגרות**

**במדעי המחשב שאלון 899381**

תכנון זמנים

משך הבחינה 3 שעות הבחינה מורכבת מ 5 שאלות ב 3 חלקים:

חלק א - יסודות - 2 שאלות - ניקוד כולל של 25 נקודות

חלק ב - מבני נתונים - 2 שאלות - ניקוד כולל של 50 נקודות

חלק ג - יחידה 5 - שאלה אחת - 25 נקודות

**חלוקת זמנים מומלצת -**

פרק א - 40 דקות

פרק ב - 80 דקות

פרק ג - 40 דקות

סך הכל - 160 דקות לפתרון שאלות

20 דקות נוספות יוקדשו למעבר על הבחינה, בחירת השאלות, ובדיקה של התשובות בחלוקה בין הפרקים השונים.

כללי כתיבה נכונה

שמות משמעותיים למשתנים ופעולות, לשמור על שמות קצרים יחסית על מנת לא לבזבז זמן על כתיבת שמות ארוכים מדי.

* שמות משתנים מתחילים באות קטנה (ואות גדולה בכל מילה נוספת)
* שמות פעולות מתחילים באות קטנה ואם מורכב משתי מילים יש להתחיל את המילה השנייה באות גדולה.
* בכל זימון של פעולה שלא מקבלת פרמטרים חייבים להקפיד על סוגריים.
* בכל פעולה שכותבים חייבים לכתוב טענת כניסה ויציאה
* תיעוד- יש לתעד כל קטע קוד משמעותי או חישוב מורכב

הזחה, וסוגריים מסולסלים - לשים לב שקטעי הקוד מוסטים ימינה בכמה מילימטרים, בהתאמה למבנה הקוד, ושיש שימוש תקין בסוגריים מסולסלים. בקטעי קוד ארוכים מומלץ לכתוב יעוד על הסוגר הסוגר ולתאר את איזה קטע קוד הוא סוגר.

מודולוריות - תכננו את הפתרון היטב, כדאי לכתוב שלד פתרון, לנסות להריץ אותו ובהתאם לכתוב את הפתרון המלא. השקעת זמן בתכנון טוב תקצר את זמן הכתיבה בפועל.  
במידה ויש פתרון שדורש הרבה כתיבה מומלץ לזהות משימות שעבורן יכתבו פעולות עזר.

בדיקה ובקרה - בדקו את הקוד על מגוון קלטים אשר אמורים לתת תוצאות שונות. כדאי לבדוק מקרי קצה. השקעה בבדיקה תאפשר לכם לגלות טעויות מיותרות שלא שמתם לב אליהן בכתיבה.

בכל פעולה שמחזירה ערך, חשוב לשים לב שבכל המסלולים מחזירים ערך (בדיקות התנאי בהן מוחזר ערך) - למשל, במקרה של if אחרי if ללא else וללא return אחד כללי הפעולה לא תחזיר ערך גם אם הלוגיקה נכונה ואין מקרים נוספים.  
למשל, אם נכתוב:

If num>1

Return 1;

If num<1

Return 2;

If num==1

Return 3;

למרות שמבחינה לוגית ברור שאלה שלושת האופציות היחידות, הפעולה לא תעבור קומפילציה, חייבים לעבוד עם if-else או להגדיר return אחד כללי ללא תנאי

טבלאות מעקב -

יש לבנות טבלה המכילה עמודה לכל משתנה, לתנאים העיקריים ולפלט.

חשוב להציג במעקב פלט מדויק כפי שמוגדר בפקודת הפלט ( בדיוק מה שיוצג על המסך!)

במעקב עצמים חשוב לא לצייר את העצם יותר מפעם אחת. יש להציג את השינוי בעקבות כל פעולה בתוך העצם שיצרתם

כאשר עושים מעקב על מבנה נתונים כמו רשימה/מחסנית/תור - חשוב לצייר את מצב המבנה בכל מעבר

מערכים

כאשר נדרשים לבנות מערך מונים לשים לב לפניה לאיברי המערך ולגישה לעדכון התא הנכון.  
חשוב לחשוב אם עובדים מהתא במקום ה 0 או מהתא במקום ה -1

בכל פעולה הסורקת/מעבדת מערך ומחזירה ערך חשוב לבדוק היטב האם הערך המוחזר הוא מספר התא או ערך התא.

בפניה לעצם במערך (ובכל מבנה נתונים אחר) - חשוב לוודא שהעצם קיים לפני שניגשים לבצע פעולות/לבדוק תכונות/לעדכן - לא לנסות להפעיל פעולות מתוך הממשק על עצם שלא קיים (תא במערך שהוא null)

בכל מבנה נתונים, ובעיקר במערך של עצמים לא לסרוק את כל המערך (המבנה) אלא רק כל עוד יש עצמים בתאי המערך

רקורסיה

רקורסיה על עצים - פעולה שמחזירה ערך להתחיל מתנאי העצירה

רשימה מקושרת

בכל סריקה/עיבוד/טיפול ברשימה יש לעבוד עם מצביע עזר על מנת לא לאבד את המצביע המקורי לראש הרשימה

בסריקה של רשימה יש לשים לב להבדל בין מקרים בהם סורקים עד לסוף הרשימה (עד שמגיעים ל null) או עד לאיבר האחרון (חשוב למקרים בהם בודקים למשל אם רשימה ממוינת בסדר עולה, במידה ונבצע בדיקה אם האיבר האחרון קטן מהאיבר הבא אחריו - ננסה לבדוק איבר שלא קיים ותתקבל שגיאת זמן ריצה)

לפני שבודקים ערך של חוליה יש לבדוק שהיא אינה ריקה

הכנסת איבר - חשוב לדעת לטפל במקרים שבהם צריך להכניס איבר לתחילת / אמצע/ סוף הרשימה

בפעולת מחיקה ברשימה - יש לבצע טיפול נפרד למחיקה במקרה שהאיבר (או האיברים למחיקה) הם בתחילת הרשימה, וטיפול נפרד למקרה שהאיבר באמצע הרשימה.  
על מנת למחק יש לקדם מצביע עד לחוליה הקודמת לה.  
בסיום המחיקה יש לשחרר את החוליה שנמחקה על ידי הצבעה ל null

ברשימה של עצמים - על מנת לטפל בעצם יש לגשת לעצם בעזרת פעולת האחזור של חוליה ()GetValue ורק לאחר מכן ניתן לגשת לאחזור/עדכון תכונות העצם עצמו

רשימה דו כיוונית - מתבססת על מבנה חוליות של עץ בינארי - ניתן ללכת עד הסוף ולחזור אחורה להתחלה. חשוב לשמור תמיד מצביע לתחילת הרשימה.

מחלקה המכילה תכונה שהיא רשימה מקושרת (למשל כיתה המכילה רשימת תלמידים) - אם ניתן כדאי להגדיר 2 תכונות במחלקה - מצביע לתחילת הרשימה ומצביע לסוף הרשימה.  
מצביע לסוף יכול להקל על פעולות הוספה לפי סדר הקליטה.  
יש לשמור שהמצביע לסוף אכן יצביע לסוף בסיום כל פעולת הוספה

פעולות גנריות שכדאי להכיר ולהביא אתכם:

* מספר חוליות בשרשרת
* האם ערך קיים ברשימה
* פעולות הכנסה (התחלה/סוף)
* מחיקה של איבר (חשוב לשים לב שהפעולה הגנרית מכילה התייחסות לאיבר ראשון/אמצע)
* מציאת מינימום ומקסימום

תור

גם במחסנית וגם בתור במומלץ להעתיק את המחסנית או התור ולעבוד על העותק. ככה גם אם מעוניין להרוס את המחסנית או לסיים פעולה תוך כדי ביצוע , לדוגמה בדיקה אם קיים ערך או לא. אין בעיה כי התור/המחסנית המקורית נשמרת בצד.

כאשר עובדים עם תור, אם לא רוצים להעתיק ניתן להשתמש בזקיף/ערך חריג.  
חשוב לוודא שהערך החריג לגיטימי (למשל בתור של מספרים שלמים חיוביים נוכל להשתמש ב 999-).  
חשוב לזכור להוציא את הערך החריג מהתור לאחר המעבר

כאשר צריכים להחזיר תור/מחסנית ברקורסיה, יש לשים לב שהמחסנית/התור לא מאותחלים כל פעם בפעולה הרקורסיבית, במקרה הצורך ניתן להשתמש בפעולת מעטפת (לבנות פעם אחת את התור/.מחסנית) וכך האיברים יוכנסו בכל פעם בלי שזה יאותחל מחדש.

מחסנית

חשוב לזכור שהעברה ממחסנית אחת לאחרת הופכת את סדר האיברים.  
דבר זה הופך את המבנה לנוח לבדיקת פלינדרום/תהליכי היפוך מחרוזות וכו

עצים

סוגי סריקות (תחילית, תוכית, סופית) - להכיר את הקשר בינהן, איפה נמצא שורש העץ בכל סריקה, מתי הן זהות. חשוב לדעת לעבור מפלט של סריקה חזרה למבנה של עץ.

כאשר הפעולה דורשת מערב על עץ ללא החזרה של ערך - פעולת void - ניתן להתבסס על אחת משלושת הסריקות (למשל - פעולות הדפסה)

תנאי עצירה (אחד או יותר). בדרך כלל null או (IsLeaf(bt או שניהם  
בדיקת ערך מוחזר עבור תנאי העצירה null או (IsLeaf(bt לא תמיד מחזירים true!

בפעולות בוליאניות, שלילה, לרוב, קלה יותר למימוש  
בפעולות בוליאניות בדיקת קשר לוגי (|| , &&) בין שני תתי העצים, בהתאם לדרישות השאלה  
לשים לב להבדל בין פעולות "האם קיים איבר שמקיים תנאי" לבין פעולות "האם כל העץ מקיים תנאי"

לעיתים יש צורך להשתמש בפעולת מעטפת - נכתוב פעולה לפי הדרישות כפעולת מעטפת ונזמן פעולה שתבצע את העיבוד/בדיקה הנדרשת הפעולה הפנימית יכולה לקבל פרמטרים לפי הצורך

פעולות גנריות שימושיות: האם עלה?, האם קיים בעץ?, מספר צמתים/עלים בעץ, רמה של עץ, צומת מינימלית/מקסימלית בעץ, תבנית מנייה/סכום/בוליאני, מציאת מסלול משורש לעלה  
חיפוש בינארי, הכנסת איבר לעץ חיפוש בינארי,

יעילות

להתחיל את התשובה תמיד מהגדרת n - אורך הקלט.  
במידה ובפתרון המוצע ישנן תתי פעולות יש לחשב יעילות עבור כל אחת מהן, חשוב לנמק בדיוק מדוע זו היעילות.  
לאחר שהוגדרו תתי הפעולות יש להגדיר יעילות לפעולה המרכזית בהסתמך על תתי הפעולות.  
עבור קטע קוד גדול אשר לא חולק לתת פעולות, מומלץ להגדיר תתי משימות עיקריות ולנתח יעילות עבורן ומשם להתקדם ליעילות של הפעולה כולה.  
יעילות ללא נימוק לא מקבלת ניקוד

לולאה מקוננת לא בהכרח מחייבת סיבוכיות ריבועית, לפעמים המעבר הוא סידרתי/לינארי - חשוב לשים לב כמה פעמים מטפלים בכל איבר בקלט

יחידה 5 - תכנות מונחה עצמים

הורשה -

כל המחלקות יורשות מ- Object  
כאשר ממשמשים פעולה בונה במחלקה היורשת ממחלקה אחרת יש להגדיר את הזימון של הפעולה הבונה של מחלקת הבסיס (super)

חשוב להבחין בין הרשאות גישה של תכונות ופעולות public / protected / private

הורשה של משתנה סטטי (משתנה מחלקה) - נוצר פעם אחת במחלקת הבסיס  
אין לצייר את המשתנה הסטאטי בתוך העצם אלא כללי לכל המחלקה! כל שינוי של התכונה הסטאטית תשנה את ערכו עבור כל העצמים במחלקה.

העמסת פעולות - פעולות באותה מחלקה - בעלות שם וערך מוחזר זהה אך שונות בסוג/מספר הפרמטרים שהן מקבלות)

עקרון הדריסה – כתיבת פעולה בעלת חותמת ( כותרת) זהה ( שם, ערך מוחזר ופרמטרים מתקבלים זהים) במחלקת האב ובמחלקות היורשות.

עקרון הדריסה יאפשר גישה לפעולה מהמקום בה נוצרה ולא מצורת ההסתכלות עליה.

פולימורפיזם - פולימורפיזם מתבסס על היכולת להסתכל על אובייקט בצורות שונות ולהמיר אובייקטים מטיפוס לטיפוס.

* מי שקובע אלו מהפעולות ניתן להפעיל זה טיפוס ההפניה (רק פעולות המוגדרות במחלקה של טיפוס ההפניה)
* מי שקובע איזו פעולה תופעל זה טיפוס העצם (אבל רק פעולות שטיפוס ההפניה מכיר).

כאשר יוצרים מערך פולימורפי  - לכל אברי המערך יש מפנה מאותו סוג, אבל חשוב מאד להבחין עבור כל עצם שיוצרים מאיזה מחלקה הוא נוצר. כלומר – ההסתכלות על המערך היא מכיוון ההפניה אבל העצם נוצר לפי הבנאי שהפעלנו.

במעקב אחר קוד שעובר על עצמים שיש בהם מרכיב פולימורפי חשוב לבדוק היטב עבור כל פעולה:

* מהו טיפוס המפנה ( איך מסתכלים על העצם כרגע?)
* מהו טיפוס העצם ( מאיזה בנאי הוא נוצר?)
* לוודא מאיזה מחלקה תופעל הפעולה (האם היא מוגדרת גם במחלקת הבסיס וגם במחלקה היורשת – עקרון הדריסה )

להקפיד במעקב להשתמש בתרשים עצמים/מפת זכרון - כדי לא לטעות.   
  
המרות -

לזכור שההמרות מתבצעות בזמן ריצה ולא בשלב הקומפילציה.

המרה כלפי מעלה – המרה בטוחה תמיד.   
לאחר שנעשתה המרה כלפי מעלה, אם רוצים לזמן שיטה מהמחלקה המקורית - יש לבצע המרה כלפי מטה.

המרה כלפי מטה – לא בטוחה, לכן נדרשת מפורשת.  
לדוגמא:

Animal rob=new Animal “)Rob","female20,");

(Mammal(rob)). nurseFrom(mom) //ILLEGAL – Run Time Error

המרות מתבצעות בזמן ריצה ולא בזמן קומפילציה ולכן המהדר לא יזהה שגיאה בהמרה זו, כיוון שבוצעה המרה מפורשת.   
בזמן ריצה יתברר כי rob  נוצר כ'בעל חיים' ולכן לא ניתן להמירו ל'יונק'

בהמרה למטה – חובה לבדוק היתכנות שלה (על ידי instanceof)

ממשקים

מחויבות מחלקה המממשת ממשק לממש את כל הפעולות בה (באופן ישיר או עקיף)

לא ניתן ליצור עצם מממשק. אבל ניתן ליצור הפניה מהטיפוס שלו.

במימוש ממשק חובה לממש את כל השיטות שלו.

ממשק יכול לרשת מממשק!

