| סמסטר | מועד | סעיף | שאלה | תשובה |
| --- | --- | --- | --- | --- |
| 2020א | 86 | א | הקצאת זיכרון למשתנה חיצוני (גלובלי) נעשית כפעולה ראשונה, בריצת התוכנית | לא נכון. לפי עמוד 40 בספר:  “the initialization is done once only conceptionally, before the program starts executing” |
| ב | בפקודת printf , המהדר מוודא שמספר הערכים המודפסים, זהה למספר המשתנים המתואר במחרוזת הפקודה. | לא נכון. המהדר אינו מוודא זאת אך עלולים לקבל שגיאות בהדפסה. נעשית בדיקה על ידי הפונקציה עצמה .  הקומפיילר רק מוודא שנשלח ערך כלשהו (יכול להיות גם NULL). אם לא ישלח ערך הוא ידפיס ערך כלשהו. |
| ג | לא ניתן לקרוא לקריאות מערכת ההפעלה UNIX ,מתוך התוכנית הראשית. יש לכתוב פונקציית ספריה מיוחדת, אשר תבצע את הקריאה. | לא נכון. רשום בע"מ 169 כי מערכת UNIX מפתחת שירותים על ידי מגוון קריאות מערכת, אשר הופנו במערכת ההפעלה אשר נקראות ע"י תוכנות המשתמש .  ישנן מגוון קריאות אשר מאפשרות ללא כתיבת ספרייה מיוחדת, כמו למשל write , read . |
| מועד ב | א | פונקציה מקבלת תמיד מספר ידוע של ארגומנטים. | לא נכון, ישנה ספריה stdarg.h שמאפשרת לקבל משתנה של פרמטרים. צריך לכול את הספרייה.  אילולא הספרייה , הטענה הייתה נכונה מכיוון שבכל חתימת פונקציה היינו יכולים לראות את מספר הארגומנטים.  למשל: הפונקציה printf . עמוד 155 בספר |
| ב | ניתן להעביר רשומה (structure) , כפרמטר לפונקציה, רק לפי כתובת. | לפעמים. לפי עמוד 129 בספר:  There are at least three possible approaches: pass components separately’ pass an entire structure, or pass a pointer to it  לפי מה שכתוב בספר ,ניתן להעביר רשומות בדרכים נוספות מלבד לפי כתובת. אכן ניתן להעביר את כל הרשומה על המחסנית. אבל הנ"ל אינו מומלץ -משיקולי מקום במחסנית. |
| 2019ב | 87 | א | ניתן לצאת, תמיד, מתוך כל לולאת for , על ידי הפקודה break | לפעמים. במידה ואנחנו בתוך לולאה מקוננת מכל סוג או switch case , הbreak יוציא מאותה לולאה או switch. בכל מקרה אחר, אנו נצא מאותה לולאת for. |
| ב | בביטוי הבא :  a = 10;  b = 3;  c = a---b;  מתבצע בעצם :  --b a- כלומר ראשית ערכו של b יקטן ב1 ורק לאחר מכן תבוצע פעולת חיסור בין a לb. כלומר: תתקבל התוצאה 8 בc, ערכו של b יהיה 2 ואילו a יישאר ללא שינוי. | לא נכון . בעמוד 53 בטבלת סדר עדיפות רואים שמינוס כפול עדיף על מינוס רגיל . לכן קודם a ירד ב1 ואז יבוצע חיסור ( 9-3=6). |
| ג | תכונותיה של תיקיה (directory) , במערכת UNIX, שונה מתכונותיה של קובץ רגיל . כלומר תכונות התיקיה כגון שם, גודל ,תאריך יצירה וכו' שונות מתכונות קובץ | לא נכון. במערכת UNIX תיקיות נחשבות כקבצים לכל דבר ולכן מגיעות עם אותן תכונות כקבצים . |
| 2019א | 85 | א | משתנה המוגדר מחוץ לפונקציה הוא למעשה גלובלי, ומוכר בכל מקום בתוכנית | לפעמים. אם המשתנה הוגדר בראש התוכנית, אז הוא מוכל לכל הפונקציות. למעשה, המשתמש הנ"ל מוכר רק מהמקום שהוא הוגדר ומטה וכל מי שמעליו לא "יראה" אותו. אם יש צורך להשתמש במשתנה הנ"ל בקבצים אחרים של התוכנית, יש לרשום "external" ואז שם המשתנה בקובץ (קובץ C) על מנת שאותן הפונקציות באותו הקובץ יוכלו לגשת אליו |
| ב | המשפט הבא תקין :  i =101;  a[i]=i++; | לא נכון. המשפט אינו תקין ואינו חד משמעות. השפה לא נובעת עם סדר קידום של i לכן עלולים לקבל תוצאות בלתי צפויות (עמוד 53 בספר) |
| ג | לא ניתן לפתוח יותר ממספר קבוע, מקסימלי, של קבצים בתוכנית אחת. לכן יש לאחד קבצים מראש, או להשתמש בכמה תוכניות | ע"מ 162 :  “since most operating systems have some limit on the number of files that a program may have open simultaneously, it’s a good idea to free files pointers…”  לפעמים. הוא תלוי במכונה, אך לפי ע"מ 162 לרוב המכונות יש הגבלה לכמות הקבצים שניתן לפתוח. |
| 82 | א | משתנה חיצוני external, מוגדר במפורש מחוץ לפונקציות המשתמשות בו, לכן אין צורך להצהיר עליו שוב בפונקציה המשתמשת | לפעמים נכון ולפעמים לא נכון.  אם המשתנה החיצוני מוגדר אחרי הפונקציה שמשתמשת בו או שהוא מוגדר בקובץ אחר, יש להצהיר עליו שוב בתוך הפונקציה בתור extern (extern int i) |
| ב | אין חשיבות לסדר הפונקציות בתוכניות | לפעמים. פונקציה מוכרת מאיפה שהיא כתובה והלאה מכאן שהפונקציה יכולה להשתמש רק בפונקציה שכתובה לפניה אבל אפשר למנוע זאת אם נצהיר על הפונקציה בתחילת התוכנית |
| ג | הפורמט והתוכן של תיקיה (directory) זהים בכל מערכת הפעלה, לפיכך ניתן לכתוב פונקציית חיפוש לקובץ, בתוך תיקיה , ולהשתמש בפונקציה זו ב- UNIX ובכל מערכת הפעלה אחרת | לא נכון. לכל מערכת הפעלה יכול להיות מוסכמה אחרת לגבי ייצוג של הרעיות "תיקיה" (למשל בUNIX היא קובץ המכיל מידע אודות הקבצים באותה תיקיה) עמוד 171 בספר.  עם זאת ,באמצעות קריאות מערכת (בהנחה שכל המערכות הרצויות אכן ממשות את הsyscall הדרושים) ניתן ליצור פונקציה שתציג מימוש כלפי המתכנת |
| 2018ב | 85 | א | לאחר ביצוע המרה (cast) למשתנה, ניתן, מאותו מקום והלאה, להתייחס למשתנה כאילו הוא הוגדר מהטיפוס אליו בוצע ההמרה.  לדוגמא: אם i הוא מטיפוס int , אז לאחר השורה c=(char)i המהדר יכיר בכל המופיעים של i כ-char לכל דבר, ואין צורך בהמרות נוספות של i. | לא נכון. לפי עמוד 205 בספר:  “A unary expression preceded by the parenthesized name of a type causes conversion of the  value of the expression to the named type.”  כלומר בדוגמא שיש בשאלה ה iעצמו לא משתנה ,אלא הערך i מקבל את ערכו ה"char" של i וכך זה גם ב0 פעולת cast .  הערך שעבר cast לא משנה את סוגו , אלא רק באופן זמני |
| ב | פונקציה המחזירה תמיד ערך, אלא אם מוצהר שהערך החוזר הוא void. | נכון. פונקציה מחזירה ערך כלשהו , אם בפונקציה לא מוחזר ערך החזרה, כלומר מה הסוג משתנה שהיא מחזירה, הברירת מחדל היא שהוא יחזיר int , אחרת כל פונקציה מחזירה ערך כלשהו , פרט לפונקציות מסוג void .  זוהי בחירה של המשתמש אם להשתמש בערך המוחזר או לא . |
| ג | במערכת UNIX ניתן לקרוא מקובץ, בסדר כלשהו, מבלי לקרוא סדרתית את כל התווים שבו | נכון. לפי עמוד 174 בספר : מדובר על הפונקציה Iseek :  " When necessary, however, a file can be read or written in any  arbitrary order.”  הפונקציה מאפשרת שיטוט בקובץ באיזה סדר שנרצה ולכן הטענה נכונה |
| 2018א | 91 | א | משתנה סטטי מאותחל, שהוגדר כCONST,זהה לערך קבוע, שהוגדר על ידי #define  לדוגמה:  Static const int N=42;  זהה ל:  #define N((int)42) | לפעמים. הטענה איננה נכונה כיוון שבעת תהליך הקדם מעבד נפרסות ההצהרות בdefine והאות N  מה שלא מתרחש עבור const סטטי מאותחל.  בנוסף , סטטיק const נוצר כמשתנה בסטאק ונשאר שם לאורך כל חיי התוכנית כך שגם מבחינת זיכרון אין לזה שקול.  עם זאת , האענה נכונה מבחינת חלק מהשימושים כך שניתן להגדיר משתנה בצורה זאת או זאת ועבור המשתמש הפשוט השימוש דומה, ושניהם ייצגו את המספר 42. |
| ב | ניתן להשתמש באינדקס שלילי במערך. לדוגמא : x[-3] הוא ביטוי חוקי בשפה | נכון. הביטוי x[-3] הינו ביטוי חוקי, ויחזיר את הערך במקום ה x-3 שכנראה יכיל ערך זבל כלשהו (לא חייב . באחריות הכותב) , אולם לא יגרום להתראה על שגיאה. |
| ג | אין להשתמש בקריאת המערכת “creat” עבור קובץ קיים | לא נכון. על פי עמוד 173 ניתן לקרוא באמצעות creat לקובץ קיים:  “If the file already exists, creat will truncate it to zero length, thereby discarding its previous contents”  בנוסף רשום במפורש  “It’s not an error to creat a file that already exists” |
| 2017ב | 87 | א | יש לאתחל משתנים אוטומטיים , בשל הגדרתם. | לא נכון. כל משתנה שבתוך פונקציה מוגדר אוטומטית כAUTO וכאמור ניתן לבצע  Int c;  (ללא אתחול) אך לא ניתן להשתמש במשתנה C, עד שיאותחל מפני שלא יודעים מה ערכו (ערך זבל) ולשם תכנות נכון יש לאתחלו כבר בהגדרה. |
| ב | ניתן לבצע אריתמטיקה של מצביעים. למשל, אפשר לבצע כפל ב-2 של מצביע למערך, בכדי להגיע לאיבר אחר במערך | לפעמים. תלוי איזה סוג פעולה מבצעים , למשל חיסור מצביעים הוא חוקי , אך לפי עמוד 138 חיבור ביניהם לא. לכן גם פעולת הכפל לא שכן כפולה של פוינטר במספר 'X' הינה תוצאת החיבור של הפוינטר עם עצמו 'X' פעמים. |
| ג | לא ניתן לקרוא לקריאות מערכת ההפעלה UNIX, מתוך התוכנית הראשית. יש לכתוב פונקציה ספריה מיוחדת, אשר תבצע את הקריאה. | לא נכון. ניתן לקרוא לפונקציות מערכת מתוך התוכנית הראשית. |
| 85 | א | אם הפונקציה FUNC מופיעה באב טיפוס (prototype ) באופן הבא:  Double func(double)  אז בקריאה לפונקציה :  X= func (3);  המספר 3 (שהוא integer ) יהפוך לערך 3.0 (שהוא double ) ללא צורך בהמרה (cast) . | נכון. במצב כזה מתבצע casting אוטומטי ללא צורך בהמרה מפורשת |
| ב | משתנים חיצוניים מוגדרים מחוץ לפונקציות, ולכן מוכרים על ידי כל הפונקציות בתוכנית. | לפעמים. משתנה חיצוני מוכל לכל הפונקציות שנמצאות תחתיו/אחריו בקוד.  ולכן אם הוא מוגדר מעל כל הפונקציות הוא יוכל בכל הפונקציות.  אחרת, יוכר רק באלה שנמצאות תחתיו |
| ג | במערכת UNIX לא ניתן לקבוע מראש את הרשאות הקובץ. למשל, לא ניתן ליצור קובץ לקריא בלבד , יש לייצרו ורק אח"כ לשנות את ההרשאה | לא נכון. עמוד 173 זה בדיוק מה שעושה creat:  Fd=creat (name, perms);  מייצרת קובץ כאשרPERNS הוא :  “permissions specified by the perns arguments.” |
| 2017א | 86 | א | פניה למקום ה10 במערך באורך 9,תגרור שגיאה בזמן הידור | לא נכון. לא נקבל שגיאת הידור, אך עלולים לקבל שגיאת ריצה |
| ב | אין חשיבות לערכו של EOF, כל עוד הוא שונה מערכם של כל התווים האחרים במערכת | נכון, EOF בהגדרה חייב להיות שונה מכל משתנה מסוג CHAR, ולכן הוא מוגדר כ (1-) שהוא מספר שלא מייצג אף אות בASCII |
| ג | שימוש בקריאת המערכת ISEEK מאפשרת שיטוט בקובץ. אך כתוצאה משימוש בה , עלול להשתנות תוכן הקובץ | לא נכון, ISEEK מחזירה מצביע למיקום כקובץ שאנחנו מעוניינים, אך אם חלה תקלה הפונקציה פשוט תחזיר (1-). |
| 84 | א | התוכנית הראשית (main) חייבת להחזיר ערך לסביבת העבודה. זאת בכדי להעביר לסביבה סטטוס הצלחה/כישלון. | לא נכון. פונקציית הmain יכולה להיות פונקציית void. |
| ב | שדה סיביות (bit field) יכול להיות בעל 8 שדות לכל היותר. | לא נכון |
| ג | תכניות המשתמשות באופן ישיר בקריאות מערכת ההפעלה הן ניידות (portable). כלומר, ניתן להריצן במערכת הפעלה אחרת, ללא התאמה למערכת, בדומה לשימוש בספריה הסטנדרטית. | לפעמים |
| 2016ב | 90 | א | במקרה של מספרים מסוג integer, פעולת & (and ביטים) ופעולת && (and לוגי) הן למעשה שקולות. | לפעמים נכון ולפעמים לא. פעולת & היא פעולת השוואה בין סיביות של שני מספרים, בעוד ש && זהו ביטוי לוגי שמחזיר 1 או 0. במקרים מסוימים כמו במספרים 1 ו-0 נקבל את אותה התוצאה בין שני הפעולות &, &&. במקרים אחרים, לדוגמה שנשווה בין המספרים 2, עבור הפעולה & נקבל שהתשובה היא 2 (כי זה משווים בין הביטים שדלוקים במקום השני) אבל עבור && נקבל 1 כלומר true ששני הערכים זהים. |
| ב | כאשר משתנה אוטומטי (מקומי לפונקציה) מוגדר כ-static, הוא הופך למעשה למשתנה גלובלי. | לא נכון. משתנה לוקלי שהוא סטטי אומנם נשמר ב data ולא במחסנית אך אין גישה אליו מחוץ לבלוק הפונקציה בה הוא מוגדר, ולכן אינו זהה למשתנה חיצוני. |
| ג | קובץ אשר נפתח על ידי fopen, לא נסגר על ידי מערכת ההפעלה, בסיום התכנית. לכן יש צורך בשימוש ב-fclose, לשם סגירתו. | לא נכון. עמוד 162 בספר: fclose נקרא באופן אוטומטי בסיום כל תוכנית ע"י המערכת. עם זאת, מומלץ לסגור את הקובץ בסיום השימוש. |
| 87 | א | אם לאופרטור אריתמטי יש שני אופרנדים: אחד מסוג int והשני מסוג float, אזי האופרנד מטיפוס float יומר לטיפוס int. | לא נכון, לפי עמוד 44 בספר, אם אחד הdata types הוא float אז type הנמוך יותר יקודם להיות גם הוא מסוג float. |
| ב | הפונקציה main נקראת בתחילת התכנית, אולם ניתן לקרוא גם לה מתוך פונקציות אחרות, כפי שניתן לקרוא לכל פונקציה אחרת בתכנית. | נכון, אם כי זה מקרה פחות נפוץ. ניתן לקרוא לה מכל פונקציה אחרת לפי ההגדרות הרגילות של C. |
| ג | תכנית אחת יכולה לעדכן לכל היותר מספר מוגדר מראש (מקסימאלי) של קבצים. | לפעמים נכון ולפעמים לא. אין הגבלה לגבי מספר הקבצים שניתן לעדכן, אולם יש הגבלה על מספר הקצבים שתוכנית יכולה לפתוח במקביל ובנוסף ישנה הגבלה על מס' התוכניות שיכולות לפתוח את אותו הקובץ במקביל. |
| 2016א | 93 | א | שימוש במצביע (pointer) לבלוק זיכרון שקול, למעשה, לשימוש במערך (array). |  |
| ב | לאחר שהוגדר אב הטיפוס של פונקציה (prototype), יש להעביר בקריאה לפונקציה, מספר ידוע של ארגומנטים, כמספר שנקבע בהגדרת אב הטיפוס. | נכון, אנו נקבל שגיאה אם לא נקרא לפוקנציה בהתאם לגדרת האב טיפוס (החתימה של הפונקציה). |
| ג | הפורמט והתוכן של תיקיה (directory) זהים בכל מערכת הפעלה. לפיכך ניתן לכתוב פונקציית חיפוש לקובץ, בתוך תיקיה, ולהשתמש בפונקציה זו ב-UNIX ובכל מערכת הפעלה אחרת. |  |
| 86 | א | בכל פעם שהמהדר מטפל במחרוזת קבועה, הוא מקצה מקום, בגודל השווה לאורך המחרוזת + 1, בחלק ה-data של התכנית. | נכון, התו האחרון מצביע על סיום המחרוזת ‘\0’ |
| ב | **ערכו** של משתנה a מסוג מערך (array) הוא **הכתובת** של האלמנט הראשון במערך a[0]. | נכון, הוא מצביע על האיבר הראשון ברשימה, עמוד 28. |
| ג | בכדי לקרוא בתים ניתן להשתמש ב-system call (קריאת מערכת) "read". מומלץ לקרוא את כל הבתים יחדיו, ולא לקרוא בית אחד בכל קריאה. | נכון, כי זה יעיל יותר שיש פחות קריאות מערכת ברגע ששולחים כמה ביחד ולא אחד אחד. עמוד 170. |
| 84 | א | הקצאת זיכון למשתנה חיצוני (גלובלי) נעשית כפעולה ראשונה, בריצת התכנית. | לא נכון, הקצאת זיכרון למשתנים לא נעשית במסגרת ריצת התוכנית אלא בתהליכים הקודמים לה. |
| ב | הצבת הקבוע 0 לתוך מצביע, היא שגיאה, מכיוון ש-0, אינו מען חוקי עבור נתונים. | לא נכון, הצבת הקבוע 0 זהה להצבת הקבוע NULL שערכו שווה ל0. |
| ג | ניתן להשתמש במשתנה מסוג unsigned short בבטחה, מכיוון שגודלו זהה תמיד, בכל מערכת. | לא נכון, תקן ANSI-C לא מחייב מה יהיה גודל של קבוע זה, ייתכן שיהיה שונות בין מערכות שונות, אך בפועל ברוב מערכות ההפעלה גודלו זהה. |
| 2015ב | 83 | א | אין חשיבות לסדר הפונקציות בתוכנית. | לפעמים, במידה ולא מתבצע שינוי על משתנים המשותפים לכמה פונקציות. לדוגמה אם נקרא לפונקציות שמבצעות פעולות אריתמטיות נקבל הבדל. כמו ההבדל בסדר פעולות חשבון. |
| ב | בפקודה printf, המהדר מוודא שמספר הערכים המודפסים, זהה למספר המשתנים המתואר במחרוזת הפקודה. | לא נכון. הקומפלייר רק מוודא שנשלח ערך כלשהו (יכול להיות גם NULL). אם לא ישלח ערך הוא ידפיס ערך כלשהו. |
| ג | אין להשתמש בקריאת המערכת "creat" עבור קובץ קיים. | לא נכון, עמוד 173 בספר כן ניתן להשתמש. |
| 2015א | 87 | א | ההחלטה איזה סוג של חילוק יתבצע, חילוק בין מספרים ממשיים או בין מספרים שלמים, מתבצעת על-פי ההקשר (context), שבו מופיע החילוק. | נכון, לפי הסוג של המשתנים אותם מחלקים. |
| ב | ניתן לקרוא, במפורש, את EOF לתוך משתנה, ללא שגיאת ריצה. | לפעמים |
| ג | במערכת UNIX ניתן לקרוא מקובץ, בסדר כלשהו, מבלי לקרוא סדרתית את כל התווים שבו. | נכון |
| 83 | א | פתיחת קובץ לא קיים לקריאה (למשל על ידי fopen) תגרום לשגיאה בזמן הריצה. | לא נכון, אם ארעה שגיאה תחזיר הפונקציה NULL. |
| ב | אם מספר האיברים באתחול רשומה (structure), קטן ממספר השדות ברשומה, האתחול נעשה לשדות הראשונים, ואילו שאר שדות הרשומה, מקבלים ערכים לא מוגדרים. | לא נכון, int יאותחלו ב0 ופונטרים יאותחלו בNULL. |
| ג | במערכת UNIX יש לכתוב ממשק (interface) מיוחד ושונה לכל קלט ולכל פלט. למשל, הממשק עבור המקלדת יהיה שונה מהממשק עבור המסך. |  |
| 2014ב | 84 | א | הקוד הבא:  int a = 0;  a++ = 5;  הוא קוד חוקי. | לא נכון, צד שמאל (המשתנה) חייב להיות מסוג שמכיל value ולא פעולה אריתמטית. |
| ב | ניתן לבצע אריתמטיקה של מצביעים. למשל, אפשר לבצע כפל ב-2 של מבציע למערך כדי להגיע לאיבר אחר במערך. | לא נכון. ניתן לבצע פעולות חיבור אך לא פעולות כפל. Char \*p = p+2; יעבור בניגוד ל char \*p = p\*2; נקבל כאן שגיאת קומפילציה. |
| ג | במערכת UNIX לא ניתן לקבוע מראש את הרשאות הקובץ. למשל, לא ניתן ליצור קובץ לקריאה בלבד, יש לייצרו ורק אחר-כך לשנות את ההרשאה. |  |
| 2013ב | 85 | א | אם הפונקציה func מופיע באב טיפוס (prototype) באופן הבא:  double func(double);  אז בקריאה לפונקציה:  x = func(3);  המספר 3 (שהוא integer) יהפוך לערך 3.0 (שהוא double) ללא צורך בהמרה (cast). | נכון |
| ב | כאשר רוצים לחסוך בזיכרון, ניתן להשתמש ב-bit-field. באופן זה אורזים כמה אובייקטים במילת זיכרון אחת. כך גם ניתן להתייחס לכתובת כל ביט, על יד שימוש באופרטור הכתובת '&'. | לא נכון |
| ג | כדי לקרוא בתים ניתן להשתמש ב-system call (קריאת מערכת) "read". מומלץ לקרוא בית אחת בכל קריאה. | לא נכון, עדיף לקרוא כמה בתים בקריאה אחת כי זה יעיל יותר שיש פחות קריאות מערכת ברגע ששולחים כמה ביחד ולא אחד אחד. עמוד 170. |

| סמסטר | מועד | סעיף | שאלה | תשובה |
| --- | --- | --- | --- | --- |
| 2013א | 84 | א | **ערכו** של משתנה a מסוג מערך (array) הוא **הכתובת** של האלמנט הראשון במערך a[0]. | נכון. בשפת c , משתנה מסוג מערך , למעשה מכיל את כתובת התא הראשון במערך, כאשר כותבים למשל a[2];  בזמן הידור הקומפיילר מחליף את זה ב: \*(a+2) כלומר 2 תאים אחרי הכתובת של a . |
| ב | משתנים חיצוניים מוגדרים מחוץ לפונקציות, ולכן מוכרים על ידי כל הפונקציות בתכנית. | לא נכון. משתנים חיצוניים מוכרים רק ממקום הצהרתן והלאה . עד סוף הקוד .  בע"מ 80 כתוב :  “The scope of an external variable or a function lasts from the point at which it is declared to the end of the file” |
| ג | הקצאת זיכרון (storage allocation) אינה תלוית מכונה, כך שניתן להעביר תכניות C ממערכת UNIX למערכת אחרת, ללא חשש. | לא נכון. הקצאת זיכרון משתנה ממכונה למכונה (machine-dependent ) ולכן התוכנית ב UNIX לא ניידות .  יש לבצע התאמות עבור סוג מערכת ההפעלה . |
| 81 | א | כאשר משתמשים באופרטור מודולוס % (modulus) עבור משתנים מסוג float, נלקח החלק השלם של כל משתנה.  לדוגמה:  float x = 20.5;  float y = 4.0;  float z;  z = x%y;  בתוכנית זו יתבצע בעצם:  20%4 ונקבל את התוצאה 5 במשתנה z. | לא נכון. עמוד 41 :  “The % operator cannot be applied to a float or double”  לא נכון מכיוון שנתקבל במקרה זה שגיאה שהרי לא ניתן להשתמש באופרטור זה על משתנה מסוג float. |
| ב | משתנה שהוגדר כ-const זהה למעשה לערך קבוע, שהוגדר על ידי #define. | לפעמים, ישנם const שניתנים לשינוי בעוד שdefine לא ניתן לשינוי בשום צורה. ניתן לאתחל את הקבועים במספרים שונים אך לא את הdefine. |
| ג | לא ניתן לקרוא לקריאות מערכת ההפעלה UNIX, מתוך התכנית הראשית. יש לכתוב פונקציית ספריה מיוחדת, אשר תבצע את הקריאה. | לא נכון. רשום בע"מ 169 כי מערכת UNIX מפתחת שירותים על ידי מגוון קריאות מערכת, אשר הופנו במערכת ההפעלה אשר נקראות ע"י תוכנות המשתמש .  ישנן מגוון קריאות אשר מאפשרות ללא כתיבת ספרייה מיוחדת, כמו למשל write , read . |
| 2012ב | 87 | א | פונקציה מחזירה תמיד ערך, אלא אם כן מוצהר שהערך החוזר הוא void. | נכון  נכון, פונקציה תמיד מחזירה ערך, ורק במצב שהיא void היא לא מחזירה ערך. |
| ב | אם מספר האיברים באתחול רשומה (structure), קטן ממספר השדות ברשומה, האתחול נעשה לשדות הראשונים, ואילו שאר השדות ברשומה, מקבלים ערכים לא מוגדרים. | לא נכון  באתחול מבנה, עם שדות לא מקבלים ערכים הם מאותחלים ב-0 |
| ג | שימוש בקריאת המערכת iseek מאפשר שיטוט בקובץ. כתוצאה מהשימוש בה עלול להשתנות תוכן הקובץ. | לא נכון  הפונקציה מאפשרת הזה של מיקום המצביע לתוך הקובץ , ולא יכול להשפיע כלל על תוכן הקובץ |
| 2011ב | 82 | א | משתנה המוגדר מחוץ לפונקציה הוא למעשה גלובלי, ומוכר בכל מקום בתכנית. | לא נכון. משתנה שהוצהר מחוץ לפונקציה מוכר רק מהמקום שהוא הוצהר ועד סוף הפונקציה. |
| ב | ניתן להעביר רשומה כפרמטר לפונקציה רק לפי כתובת. | לפעמים  במידה וארגומנטים שהפונקציה מקבלת הם מצביעים אז במילא מה שעובר זה הכתובת, אז ניתן לשלוח לפונקציה כתובת נקיה והיא תקבל את זה |
| ג | שימוש בקריאת המערכת iseek מאפשר שיטוט בקובץ. אך כתוצאה משימוש בה עלול להשתנות תוכן הקובץ. | לא נכון |
| 2010ב | 87 | א | אם הפונקציה func מופיע באב טיפוס (prototype) באופן הבא:  double func(double);  אז בקריאה לפונקציה:  x = func(3);  המספר 3 (שהוא integer) יהפוך לערך 3.0 (שהוא double) ללא צורך בהמרה (cast). | נכון, שפונקציה אמורה לקבל double והיא מקבל משתנה int, מתבצע קסטינג אוטומט ואין צורך לעשות כסטינג מפורש. |
| ב | האופרטור '&' מבצע הקצאת זיכרון עבור משתנה אשר לא בוצעה עבורו הקצאה קודמת. לדוגמה, במקרה של הפקודה: b = &a; מוקצה מקומו של משתנה b בזיכרון, ומושווה למקומו של a. | לא נכון, האופרטור & מסמן שאנחנו רוצים לעשות השמה(להעביר) של הכתובת של משתנה a למשתנה b. |
| ג | במערכת UNIX ניתן לפתוח, בו-זמנית, קבצים ללא הגבלה, על ידי שימוש בפונקציית המערכת "open". | לא נכון יש הגבלה על מספר הקבצים שניתן לפתוח (בספר כתוב לעיתים קרובות 20) |
| 2010א | 82 | א | לא ניתן להגדיר משתנה, בתוך בלוק פנימי של פונקציה (סוגריים מסולסלים פנימיים). ניתן להגדיר את כל המשתנים בתחילת הפונקציה בלבד. | לא נכון, אם כי נהוג להגדיר משתנים בתחילת הפונקציה. עמוד 9 בספר פסקה ראשונה. ניתן להגדיר משתנה גם בתוך פונקציה כל עוד מגדירים אותו לפני השימוש. |
| ב | אין חשיבות לערכו של EOF, כל עוד הוא שונה מערכם של כל התווים האחרים במערכת. | תמיד נכון. ערכו של EOF משמש לזיהוי סוף קובץ / קלט ולכן אם הוא שונה משאר התווים אין הבדל לערכו. עמוד 152 בספר שורה ראשונה. |
| ג | קובץ אשר נפתח על-ידי fopen, נסגר תמיד על-ידי מערכת ההפעלה, בסיום התוכנית. כך שאין צורך מעשי בשימוש ב-fclose. | נכון אבל לא תמיד. הקובץ ייסגר בסיום התוכנית, אבל ישנן הגבלות על פתיחה של אותו קובץ ממספר תוכניות שונות, ולכן מומלץ לסגור בסיום השימוש על מנת לא לפגוע בשימוש של תוכניות אחרות בקובץ. |
| 2009ב | 92 | א | האופרטור '&' מבצע הקצאת זיכרון עבור משתנה אשר לא בוצעה עבורו הקצאה קודמת. לדוגמה, במקרה של הפקודה: b = %a; מוקצה מקומו של משתנה b בזיכרון, ומושווה למקומו של a. | לא נכון, כשהאופרטור & לפני המשתנה זה גורם להשמת של כותבת המשתנה ולא הערך שלו, בדוגמא אנחנו עושים השמה לכתובת של a למשתנה b, מה שקורה בעצם, שהערך של b הוא הכתובת של a. |
| ב | הצבת הקבוע 0 לתוך מצביע, היא שגיאה, מכיוון ש-0, אינו מען חוקי עבור נתונים. | לא נכון, הצבת 0 לתוך מצביע גורמת לכך שהערך שלו יהיה null, וזאת אינה שגיאה, במידה וננסה לקבל את הערך במוצבע נקבל שגיאה כי הוא null והוא לא מצביע לאף מקום. |
| ג | תיקייה (Directory) שמערכת UNIX היא בעצם קובץ, ורשימת תכונותיה זהה לתכונות קובץ רגיל: שם, גודל, תאריך יצירה וכו'. | נכון, במערכת הפעלה unix, כל תיקיה היא בעצם קובץ רגיל שמכיל את כל המידע על התיקיה. |