



שאלה 1

האם ספורות המספר ממיניות בסדר עולה? — IsAscending

דרישות

- הפעולה (`IsAscending(int num)`) תחזיר `true` אם ספורותיו של `num` ממיניות בסדר עולה, אחרת `false`.
- אין להשתמש בהמרה למחוזת.
- נדרש שימוש רקורסיבי (אפשר להשתמש בעזר רקורסיבית עם פרמטר נוסף).
- אם הקלט שלילי, נבחן את הספורות של הערך המוחלט.

הסבר מלא

נפרק את המספר באמצעות פעולות חילוק ושליפה של ספרת היחידות. נשווה בין הספרה הימנית ביותר לספרה שלפניה, ונמשיך רקורסיבית על יתר הספרות.

הנחיות ליישום (אלגוריתם + קוד)

אלגוריתם

1. אם $|n| < 10$ ← החזרו `true`.

2. הוציאו שתי ספרות ימניות: $d_2 = \lfloor n/10 \rfloor \bmod 10$, $d_1 = n \bmod 10$.

3. אם $d_1 > d_2$ ← החזרו `false`.

4. המשיכו רקורסיבית עם $\lfloor n/10 \rfloor$.

קוד (C#)

```
public static bool IsAscending(int num)
{
    num = Math.Abs(num);
    if (num < 10) return true;
    int d1 = num % 10;
    int d2 = (num / 10) % 10;
    if (d2 > d1) return false;
    return IsAscending(num / 10);
}
```

שאלות הבנה וחשיבה

1. מדוע די לבדוק זוגות סמוכים מימין לשמאל?
2. מה קורה אם יש ספרות שוויה? נמקו מדוע הן מותרות/אסורות לפי ההגדרה.

דוגמאות קלט/פלט ותנאי קצה

- `IsAscending(5) → true`
- `IsAscending(1234559) → true`
- `IsAscending(132) → false`
- `IsAscending(-11239) → true`

שאלה 2

IsEven — האם מספר זוגי? (ללא שימוש ב- %)

דרישות

- הפעולה (`n`) `IsEven` תחזיר `true` אם `n` זוגי, אחרת `false`.
- אסור להשתמש באופרטור `%` (מודולו).
- הפיתרון יהיה רקורסיבי.
- תמייכה במספרים שליליים.

הסבר מלא

$$\text{isEven}(n) = \begin{cases} \text{true}, & n = 0 \\ \text{isEven}(|n| - 2), & |n| > 0 \end{cases}$$

הנחיות לישום (אלגוריתם + קוד)

אלגוריתם

1. `.true ← n = 0`
2. `.false ← n = ±1`
3. אם $0 > n$ לקרוא רקורסיבית `IsEven(n - 2)`.
4. אם $0 < n$ לקרוא רקורסיבית `IsEven(n + 2)`.

קוד (C#)

```
public static bool IsEven(int n)
{
    if (n == 0) return true;
    if (n == 1 || n == -1) return false;
```

```

if (n > 0)
    return IsEven(n - 2);
else
    return IsEven(n + 2);
}

```

שאלות הבנה וחשיבה

1. מה יקרה עבור ערכים גדולים מאוד ברכוסיה??

דוגמאות ותנאי קצר

`IsEven(1) → false , IsEven(0) → true` •

`IsEven(99999) → false , IsEven(-14) → true` •

שאלה 3

אם מספר ראשוני — IsPrime

דרישות

- הפעולה (`int num`) `IsPrime(int num)` תחזיר `true` אם `num` ראשוני, אחרת `false`.
- אין לבדוק מחלקים מעל \sqrt{n} .
- שימוש רקורסיבי בעזרת פועלות עזר הבודקת מחלק הולך וגדל.

הסבר מלא

נגדיר פעולה עזר `prime(n, d)` שבודקת שאין מחלקים בטוחה n d, \dots, \sqrt{n} .

$$\text{prime}(n, d) = \begin{cases} \text{false}, & n \leq 1 \\ \text{true}, & d > \sqrt{n} \\ \text{false}, & n \bmod d = 0 \\ \text{prime}(n, d+1), & \text{אחרת} \end{cases}$$

הנחיות לישום (אלגוריתם + קוד)

אלגוריתם

- . אם $1 \leq n$ $\text{false} \leftarrow n$.
- . בדקו מחלקים מ-2 ועד \sqrt{n} .
- . אם נמצא d כך $n \bmod d = 0$, $\text{false} \leftarrow n \bmod d = 0$, אחרת `true`.

```

public static bool IsPrime(int n)
{
    if (n <= 1) return false;
    return IsPrimeHelper(n, 2);
}
private static bool IsPrimeHelper(int n, int d)
{
    if ((long)d * d > n) return true;
    if (n % d == 0) return false;
    return IsPrimeHelper(n, d + 1);
}

```

שאלות הבנה וחשיבה

- למה מספיק לבדוק עד \sqrt{n} ?
- כיצד ניתן להאיץ: דילוג על מספרים זוגיים, טבלאות ראשוניים וכך?

דוגמאות ותנאי קצה

- `IsPrime(2) → true`, `IsPrime(1) → false`
- `IsPrime(97) → true`, `IsPrime(49) → false`

 שאלה 4**האם מספר פלינדרום (ללא המרה למחרוזת)?****דרישות**

- הפעולה (`IsPalindrome(int num)`) תחזיר `true` אם המספר פלינדרום, אחרת `false`.
- אין להמיר למחרוזת.
- מימוש רקורסיבי ע"י השוואת ספרה ראשונה לאחרונה וצמצום פנימה.

 הסבר מלא

האלגוריתם משווה בכל שלב בין הספרה הראשונה לספרה האخונה במספר. אם הן שוות ← המספר איננו פלינדרום. אם הן שוות ← קוראים רקורסיבית לחת-המספר שבאמצע (לאחר הסרת שתי הספרות הקיצונית). כאשר נותר ספרה אחת או שהמספר התרוקן לגמרי — זהו פלינדרום.

הנחיות ליישום (אלגוריתם + קוד)**אלגוריתם (השוואת קצוות)**

1. חשב את עשר בחזקת (מספר ספרות - 1) כדי להגיא למספרה הראשונה.
2. השווה בין הספרה הראשונה בספרה האخונה.
3. אם שונות ← החזר `false`.
4. אם שווות ← הסר את שתי הספרות ובדוק את המספר שנותר.
5. אם נותרת ספרה אחת או אף ספרות ← החזר `true`.

קוד (C#)

```

public static bool IsPalindrome(int num)
{
    int n = Math.Abs(num);
    return IsPalindromeHelper(n, Digits(n));
}

private static bool IsPalindromeHelper(int n, int digits)
{
    if (n < 10) return true; // ספרה אחת
    int power = (int)Math.Pow(10, digits - 1);
    int first = n / power;
    int last = n % 10;
    if (first != last) return false;
    int middle = (n % power) / 10; // הסרת ספרה ראשונה ואחרונה
    return IsPalindromeHelper(middle, digits - 2);
}

private static int Digits(int n)
{
    if (n == 0) return 1;
    return (int)Math.Floor(Math.Log10(n)) + 1;
}

```

שאלות הבנה וחשיבה

1. מדוע אנו מקטינים את מספר הספרות ב-2 בכל קרייה וקורסיבית?
2. כיצד מטפלים במקרה של מספר בעל ספרה אחת בלבד?

דוגמאות ותנאי קצה

- `IsPalindrome(0) → true` •
- `IsPalindrome(1231) → false`, `IsPalindrome(1221) → true` •

שאלה 5 **האם מספר מכיל ספרה נתונה? — HasDigit****דרישות**

- הפעולה `HasDigit(int num, int digit)` תחזיר `true` אם הספרה `digit` מופיעת ב-`num`, אחרת `false`.
- ספרה חוקית: $0 \leq d \leq 9$.
- שימוש רקורייבי, ללא המרת למחוזת.

 הסבר מלא

נשלוף בכל צעד את סכמת היחידות ונשווה `digit`; אם לא נמצאה, נמשיך עם $n/10$.

 הנחיות ליישום (אלגוריתם + קוד) **אלגוריתם**

1. אם $0 = n \leftarrow$ בדקו האם $0 = d$.

2. בדקו את סכמת היחידות: אם היא שווה ל- $d \leftarrow true$.

3. אחרת המשיכו רקורייבית עם $n/10$.

 קוד (C#)

```
public static bool HasDigit(int num, int digit)
{
    num = Math.Abs(num);

    if (digit < 0 || digit > 9) return false;

    if (num == 0) return digit == 0;
    if (num % 10 == digit) return true;
    return HasDigit(num / 10, digit);
}
```

 שאלות הבנה וחשיבות

1. כיצד משנים את הפעולה כדי להחזיר את מספר ההופעות של הספרה?

דוגמאות ותנאי קצה

- $\text{HasDigit}(0, 0) \rightarrow \text{true}$
- $\text{HasDigit}(12345, 7) \rightarrow \text{false}$
- $\text{HasDigit}(-70127, 7) \rightarrow \text{true}$

שאלה 6

GCD — המחלק המשותף הגדול ביותר (אוקלידס)

דרישות

- הפעולה (`b` int `a`, `int GCD`) תחזיר את $\text{gcd}(a, b)$ באמצעות אלגוריתם אוקלידס.
- מספרים שליליים יטופלו כערך מוחלט.
- שימוש רקורסיבי.

הסבר מלא

אלגוריתם אוקלידס מבוסס על רעיון פשוט: אם שני מספרים מתחלקים באותו מספר, גם השארית בחלוקת גדול בקטן מתחלקת בו. לכן, במקרה לחשב ישירות את $\text{gcd}(a, b)$, אנו מחשבים את $b \bmod a$ ושוב $\text{gcd}(b, a \bmod b)$, וכך עד שנגיע לקרה בסיס.

מקרה הבסיס מתרחש כאשר אחד המספרים מתאפס. במקרה זה, $\text{gcd}(a, 0) = |a|$.

$$\text{gcd}(a, b) = \begin{cases} |a|, & b = 0 \\ \text{gcd}(b, a \bmod b), & b \neq 0 \end{cases}$$

הנחיות לישום (אלגוריתם + קוד)

אלגוריתם

1. אם $0 = b \leftarrow$ החזירו $|a|$.

2. אחרת \leftarrow קראו $\text{gcd}(b, a \bmod b)$.

קוד (C#)

```
public static int GCD(int a, int b)
{
    a = Math.Abs(a); b = Math.Abs(b);
    if (b == 0) return a;
```

```
    return GCD(b, a % b);  
}
```

דוגמאות ותנאי קצה

- $\text{GCD}(18, 0) \rightarrow 18, \text{GCD}(48, 18) \rightarrow 6$
- $\text{GCD}(0, 0) \rightarrow 0$ (הגדרה מקובלת)

שאלה 7

$n!$ — עצרת! Factorial

דרישות

- הפעולה (`n` int) Factorial תחזיר $n!$ עבור $0 \leq n$.
- שימוש רקורסיבי: $1! = 1$.

הסבר מלא

$$n! = \begin{cases} 1, & n = 0 \\ n \cdot (n - 1)!, & n \geq 1 \end{cases}$$

מקרה הבסיס: $1! = 1$. צעד רקורסיבי: בכל קריאה מקטינים את n ב-1 ומכפילים בתוצאה החזרת.

הנחיות לישום (אלגוריתם + קוד)

אלגוריתם

1. אם $n < 0$ ← החזרו -1 .
2. אם $n = 0$ ← החזרו 1 .
3. אחרת ← החזרו $n \cdot \text{Factorial}(n - 1)$.

קוד (C#)

```
public static long Factorial(int n)  
{  
    if (n < 0) return -1;  
    if (n == 0) return 1;  
    return n * Factorial(n - 1);  
}
```

שאלות הבנה וחישבה

1. כיצד ניתן לכתוב גרסה איטרטיבית ל-`Factorial`?

דוגמאות ותנאי קצה

`Factorial(5) → 120, Factorial(0) → 1` •

שאלה 8

סכום ספרות של מספר — `SumDigits`

דרישות

- הפעולה `SumDigits(int num)` תחזיר את סכום כל הספרות של `num`.
- פתרון רקורסיבי; קלט שלילי יטופל בערך מוחלט.

הסבר מלא

$$\text{sum}(n) = \begin{cases} 0, & n = 0 \\ (n \bmod 10) + \text{sum}(n/10), & n \neq 0 \end{cases}$$

הנחיות ליישום (אלגוריתם + קוד)

אלגוריתם

1. אם $n = 0$ ← החזרו 0.

2. אחרת ← חשבו את ספרת היחידות ($10 \bmod n$) והוסיפו לסכום של $10 \bmod n$.

קוד (C#)

```
public static int SumDigits(int num) {  
    num = Math.Abs(num);  
    if (num == 0) return 0;  
    return (num % 10) + SumDigits(num / 10);  
}
```

שאלות הבנה וחישבה

1. כיצד מושנים לפועלה הסופרת ספרות (מספר הספרות)?

דוגמאות ותנאי קצה

`SumDigits(0) → 0` •

`SumDigits(123) → 6` •

`SumDigits(-9070) → 16` •

שאלה 9

Math.Pow — חישוב base^{exp} ללא Pow

דרישות

- הפעולה (`.exp`) תחזיר את base^{exp} עבור $0 \leq \text{exp} \leq 100$.
- שימוש רקורסיבי בסיסי.

הסבר מלא

$$\text{pow}(a, e) = \begin{cases} 1, & e = 0 \\ a \cdot \text{pow}(a, e - 1), & e > 0 \end{cases}$$

הנחיות לישום (אלגוריתם + קוד)

אלגוריתם (בסיסי)

1. אם $e = 0$ ← החזירו 1.

2. אחרת ← החזירו $a \cdot \text{pow}(a, e - 1)$.

קוד (C#)

```
public static long Power(int baseNum, int exp)
{
    if (exp < 0) return -1;
    if (exp == 0) return 1;
    return baseNum * Power(baseNum, exp - 1);
}
```

דוגמאות ותנאי קצה

- $\text{Power}(2, 10) \rightarrow 1024$, $\text{Power}(2, 0) \rightarrow 1$
- $\text{Power}(5, 1) \rightarrow 5$

שאלה 10

חישוב האיבר ה- n בסדרת פיבונacci

דרישות

- הפעולה (`n`) תחזיר את $F(n)$ לפי ההגדרה: $F(1) = 1$, $F(0) = 0$ עבור $n \geq 2$ $F(n) = F(n - 1) + F(n - 2)$

הסבר מלא

$$F(n) = \begin{cases} 0, & n = 0 \\ 1, & n = 1 \\ F(n - 1) + F(n - 2), & n \geq 2 \end{cases}$$

מקרה הבסיס: הערכים הראשונים מוגדרים ישרות $F(0) = 0$, $F(1) = 1$. צעד רקורסיבי: לכל $n \geq 2$ הפונקציה מחושבת כסכום שני איברים קודמים.

הנחיות לישום (אלגוריתם + קוד)**אלגוריתם**

1. אם $n < 0$ ← החזרו - (סדרת פיבונacci מוגדרת רק עבור $n \geq 0$).

2. אם $n = 0$ או $n = 1$ ← החזרו n .

3. אחרת ← החזרו $F(n - 1) + F(n - 2)$.

קוד (C#)

```
public static long Fibonacci(int n)
{
    if (n < 0) return -1;
    if (n <= 1) return n;
    return Fibonacci(n - 1) + Fibonacci(n - 2);
}
```

דוגמאות ותנאי קצה

$\text{Fibonacci}(10) \rightarrow 55$, $\text{Fibonacci}(1) \rightarrow 1$, $\text{Fibonacci}(0) \rightarrow 0$ •

שאלה 11**IsPalindromeString — האם מחרוזת פלינדרומית?****דרישות**

- הפעולה (`IsPalindromeString(string str)`) תחזיר `true` אם `str` פלינדרום, אחרת `false`.
- מימוש רקורסיבי; אין להשתמש במבנה עזר מיוחדים.

הסבר מלא

נשווה בין התווים הקיצוניים \leftarrow נתקדם פנימה: אם הראשון שונה מהאחרון — המחרוזת אינה פלינדרום. אחרת נקרא רקורסיבית על תזהה המחרוזת הפנימית.

הנחיות לישום (אלגוריתם + קוד)

אלגוריתם

- . $j = |s| - 1, i = 0$
- . אם $j \geq i$ true $\leftarrow i \geq j$ (המחרוזת פלינדרום).
- . אם $s[i] \neq s[j]$ false $\leftarrow s[i] \neq s[j]$.
- . אחרת \leftarrow קראו רקורסיבית עם $i + 1, j - 1$.

קוד (C#)

```
public static bool IsPalindromeString(string str)
{
    return IsPalindromeString(str, 0, str.Length - 1);
}

public static bool IsPalindromeString(string s, int i, int j)
{
    if (i >= j) return true;
    if (s[i] != s[j]) return false;
    return IsPalindromeString(s, i + 1, j - 1);
}
```

דוגמאות ותנאי קצה

- $" "$ \rightarrow true
- $"abca"$ \rightarrow false , $"aba"$ \rightarrow true

שאלה 12

היפוך מחרוזת (רקורסיבי) — **ReverseString**

דרישות

- הפעולה `ReverseString(string str)` תחזיר את המחרוזת הפוכה.
- שימוש רקורסיבי; מותר להשתמש בפועלות עזר.

הסבר מלא

בכל צעד רקורסיבי נחזיר את התו הראשון לסוף ההיפוך של שאר המחרוזת. מקרה בסיס: אם המחרוזת ריקה ← נחזיר מחרוזת ריקה.

הנחיות לישום (אלגוריתם + קוד)

אלגוריתם

1. אם $|s| \leq 1$ ← החזיר את s .
2. אחרת ← החזיר $.rev(s[1..]) + s[0]$.

קוד (C#)

```
public static string ReverseString(string str)
{
    if (str.Length <= 1) return str;
    return ReverseString(str.Substring(1)) + str[0];
}
```

דוגמאות ותנאי קצה

- $\text{ReverseString("ab")} \rightarrow "ba"$, $\text{ReverseString("")} \rightarrow ""$

שאלה 13

הסרת כל המופעים של תו נתון — RemoveChar

דרישות

- הפעולה $\text{RemoveChar(string str, char ch)}$ תחזיר מחרוזת חדשה ללא התו ch .
- פתרון רקורסיבי; אין להשתמש בלאוות.

הסבר מלא

בכל צעד רקורסיבי נבדוק את התו הראשון במחרוזת: אם הוא שווה ל- ch נדלג עליו; אחרת נצף אותו לתוצאה של קריאה רקורסיבית על שאר המחרוזת.

הנחיות לישום (אלגוריתם + קוד)

אלגוריתם

1. אם המחרוזת ריקה ← החזיר מחרוזת ריקה.
2. השוו את התו הראשון ל- ch :

3. אם שווה ← המשיכו רקורסיבית על $[1..s]$ בלי לצרף.
4. אם שונה ← צרפו את התו הראשון לתוכה של הקדימה הרקורסיבית על $[1..s]$.
5. ציריך למשוך את הפעולה `SubString` שמחזירה מחרוזת חדשה ללא התו הראשון.

קוד (C#)

```
public static string RemoveChar(string str, char ch)
{
    if (str.Length == 0) return "";
    char head = str[0];
    string tail = Substring(str);
    if (head == ch) return RemoveChar(tail, ch);
    return head + RemoveChar(tail, ch);
}
```

שאלות הבנה וחשיבה

1. כיצד תשנו את האלגוריתם שיסנן קבוצת תווים (למשל כל התנועות)?

דוגמאות ותנאי קצה

- $"" \rightarrow ('x', '')$
- $\text{RemoveChar}("banana", 'a') \rightarrow "bnn"$

שאלה 14

ספירת הופעות של תו — CountOccurrences

דרישות

- הפעולה `CountOccurrences(string str, char ch)` תחזיר את מספר ההופעות של `ch` ב-`str`.
- IMPLEMENTATION רקורסיבי; ללא לולאות.

הנחיות לישום (אלגוריתם + קוד)

אלגוריתם

אנו רוצים לספר כמה פעמים התו `ch` מופיע במחרוזת `str`. לשם כך נבדוק בכל צעד רקורסיבי את האיבר הראשון במחרוזת: אם המחרוזת ריקה – נחזיר 0. אם התו הראשון שווה ל-'`ch`' – נספר אותו כ-1 ונמשיך לבדוק את שאר המחרוזת. אם התו הראשון שונה – פשוט נמשיך לבדוק את שאר המחרוזת בלי להוסיף למספרה. בסוף, הסכימה של כל הצעדים תחזיר את מספר ההופעות הכללי.

קוד (C#)

```
public static int CountOccurrences(string str, char ch)
{
    if (str.Length == 0) return 0;

    if (str[0] == ch)
        return 1 + CountOccurrences(Substring(str), ch);
    else
        return CountOccurrences(Substring(str), ch);
}
```

דוגמאות ותנאי קצה

- `CountOccurrences("", 'a')` → 0
- `CountOccurrences("aaa", 'a')` → 3

שאלה 15**(a,e,i,o,u) — ספירת תנויות באנגלית (CountVowels)****דרישות**

- הפעולה `CountVowels(string str)` תחזיר את מספר התנויות.
- פתרון רקורסיבי; התייחסות באותיות קטנות/גדלות.

הסבר מלא

בכל צעד נבדוק אם התו הראשון שיר לקבוצת התנויות (u, o, i, e, a באותיות קטנות או גדולות), ונוסף 1 אם כן. המשיך רקורסיבית על שאר המחרוזת.

הנחיות לישום (אלגוריתם + קוד)**אלגוריתם**

1. אם המחרוזת ריקה ← החזרו 0.
2. אחרת:
3. אם התו הראשון שיר לקבוצת התנויות {u, o, i, e, a} ← הוסיפו 1.
4. המשיכו רקורסיבית על [s..1].

```

public static int CountVowels(string str)
{
    if (str.Length == 0) return 0;
    bool isVowel = "aeiouAEIOU".IndexOf(str[0]) >= 0;
    if (isVowel)
        return 1 + CountVowels(str.Substring(1));
    else
        return CountVowels(str.Substring(1));
}

```

דוגמאות ותנאי קצה

- `CountVowels("") → 0`
- `CountVowels("Hello") → 2`

שאלה 16**האם מערך של מספרים ממויין בסדר עולה? — IsSorted****דרישות**

- הפעולה (`boolean IsSorted(int[] arr)`) תחזיר `true` אם `arr` ממויין בסדר עולה.
- פתרון רקורסיבי; ללא לולאות.

הסבר מלא

בכל צעד נשווה איברים סמוכים ← נתקיים באינדקס: אם נמצא זוג יורד — המערך אינו ממויין. אם עברנו את סוף המערך בלי למצוא בעיה — המערך ממויין.

הנחיות לישום (אלגוריתם + קוד)**אלגוריתם**

1. אם $1 \leq i \leq |A|$ ← מערך ריק או בעל איבר אחד תמיד ממויין).

2. בדקו אם $A[i] \leq A[i + 1]$.

3. אם כן ← המשיכו רקורסיבית עם $i + 1$.

4. אם לא ← false (הערך אינו ממויין).

```

public static bool IsSorted(int[] arr)
{
    return IsSorted(arr, 0);
}
public static bool IsSorted(int[] a, int i)
{
    if (i >= a.Length - 1) return true;
    if (a[i] > a[i + 1]) return false;
    return IsSorted(a, i + 1);
}

```

שאלות הבנה וחשיבה

1. כיצד תתמכנו בבדיקה למערך בסדר יורד?
2. מה קורה עם `null` או מערך ריק? פרטו התנ假设ות רצiosa.

דוגמאות ותנאי קצה

- `IsSorted(new[]{}) → true`
- `IsSorted(new[]{3,2,1}) → false`, `IsSorted(new[]{1,2,2,3}) → true`

 שאלה 17**תוכנית אינטראקטיבית לבדיקת מספר פלינדרום****דרישות**

- כתבו פעולה (`int num`) `IsPalindrome` וצרו תוכנית קונסול שקוראת מספר מהמשתמש ומדפיסה הודעה מתאימה.
- אין להמיר למחרוזת; מותר להשתמש בפעולת עזר להיפוך מספר.

הסבר מלא

נשתמש בהשוואה ל-`Reverse` וכן בתוכנית קונסול.

הנחיות ליישום (אלגוריתם + קוד)**אלגוריתם**

1. קריאת מספר שלם מהמשתמש והפיכת סדר התווים בעזרת עזר `Reverse`.
2. בדיקה באמצעות `IsPalindrome`.

קוד (C#)

```

static bool IsPalindrome(int num)
{
    int n = Math.Abs(num);
    return n == Reverse(n, 0);
}

static int Reverse(int n, int acc)
{
    if (n == 0) return acc;
    return Reverse(n / 10, acc * 10 + (n % 10));
}

static void Main()
{
    Console.Write("הכנס מספר ");
    if (int.TryParse(Console.ReadLine(), out int x))
    {
        Console.WriteLine(IsPalindrome(x) ? "פלינדרום" : "לא פליינדרום");
    }
    else
    {
        Console.WriteLine("קלט לא صحيح");
    }
}

```

שאלות הבנה וחשיבות

1. כיצד לטפל במספרים גדולים מ-`int`?
2. כיצד לשלב בדיקות ייחודית לפעולה ולטפל בפתרונות (0, שלילי, מסתויים באפסים)?

 שאלה 18**המרת מספר שלם לבינארי (מחרוזת) — `DecimalToBinary`****דרישות**

- הפעולה (`n`) מ תחזיר מחרוזת המיצגת את `n` בסיס 2.
- מימוש רקורסיבי.
- אין צורך לטפל בקלט שלילי אם לא נדרש במפורש.

 הסבר מלא

$$\text{bin}(n) = \begin{cases} "0", & n = 0 \\ \text{bin}(n/2) + (n \bmod 2), & n > 0 \end{cases}$$

הנחיות לישום (אלגוריתם + קוד)

אלגוריתם

1. אם $n = 0$ ← החזר "0".
2. קרא לפעולת עזר $\text{ToBin}(n)$.
3. ב- ToBin : אם $n = 0$ ← החזר מחזורת ריקה.
4. אחרת ← קרא רקורסיבית $\text{ToBin}(n/2)$ וווסף בסוף את הספרה $n \bmod 2$.
5. התוצאה המוצברת מהקрайות תחזיר את הייצוג הבינארי המלא.

קוד (C#)

```
public static string DecimalToBinary(int n)
{
    if (n == 0) return "0";
    return ToBin(n);
}

public static string ToBin(int n)
{
    if (n == 0) return "";
    return ToBin(n / 2) + (n % 2).ToString();
}
```

דוגמאות ותנאי קצה

- $\text{DecimalToBinary}(0) \leftarrow "0"$ •
- $\text{DecimalToBinary}(13) \leftarrow "1101"$ •

שאלה 19

— חיפוש אינדקס של איבר במערך — RecursiveSearch

דרישות

- הפעולה (`RecursiveSearch(int[] arr, int target)`) תחזיר את האינדקס של `target` אם קיים, אחרת -1.
- פיתוח רקורסיבי; ללא לולאות.

בכל צעד נבדוק את האיבר הראשון שעדין לא נבדק (באינדקס i): אם הוא שווה ל- $x \leftarrow$ נחזיר את i . אם לא נתקיים רקורסיבית ל- $i+1$. אם הגיענו לסוף המערך ללא מציאה ← נחזיר -1 .

$$\text{find}(A, i, x) = \begin{cases} -1, & i = |A| \\ i, & A[i] = x \\ \text{find}(A, i + 1, x), & \text{אחרת} \end{cases}$$

הנחיות ליישום (אלגוריתם + קוד)

אלגוריתם

1. התחל ב-`.RecursiveSearch(arr, 0, target)`
2. אם i שווה לאורך המערך ← החזר -1 (המספר לא נמצא).
3. בדוק את האיבר `[i]`.
4. אם $x == a[i]$ ← החזר את i .
5. אחרת ← קרא רקורסיבית ל-`(x, i+1, target)`.

קוד (C#)

```
public static int RecursiveSearch(int[] arr, int target)
{
    return RecursiveSearch(arr, 0, target);
}

public static int RecursiveSearch(int[] a, int i, int x)
{
    if (i == a.Length) return -1;
    if (a[i] == x) return i;
    return RecursiveSearch(a, i + 1, x);
}
```

תנאי קצרה

- מערך ריק.
- מספר מופיעים — מוחזר האינדקס הראשון בלבד.
- ערכים שליליים או גדולים מאוד — נתמיכים כמו כל ערך אחר.

האם קיים מסלול במקורה דומם? — FindPath

דרישות

- הפרמטר: `grid` \rightarrow `int[][]` שבו $0 = \text{ תא חופשי}, 1 = \text{ קיר}$.
- הפעולה (`grid` \rightarrow `bool`) תחזיר `true` אם קיים מסלול מהפינה השמאלית העליונה $(0, 0)$ לפינה הימנית התחתונה $(r - 1, c - 1)$.
- מותר לזרז לעלה/מטה/שמאלה/ימינה בלבד; אין לצאת מגבולות המטריצה ואין לדרכ שוכ על תא.

הסבר מלא

נשתמש ב-DFS רקורסיבי עם מערך ביקור. בכל צעד נבדוק חוקיות תא וננסה את ארבעת הכוונים.

הנחיות לישום (אלגוריתם + קוד)

אלגוריתם

1. **בדיקה קלט:** אם המטריצה ריקה או שיש שורה ריקה \rightarrow החזרו `false`.

2. **אתחול:** הגדרו `rows = grid.Length, cols = grid[0].Length`, **מערך ביקור**, `vis[rows, cols] = false`.

3. **קריאה רקורסיבית ראשונה:** קרא ל-(`0, 0`) DFS והחזיר את תוצאתה.

4. **ב-(`r, c`):**

5. אם c, r מחוץ לתוחום \rightarrow החזרו `false`.

6. אם $1 \leq r < rows$ ו- $1 \leq c < cols$ (`grid[r][c] == true` או `vis[r, c] == true`) (`grid[r][c] == 1` (כבר ביקרנו)) \rightarrow החזרו `false`.

7. אם $1 \leq r < rows$ ו- $cols - 1 \geq c > 0$ (`grid[r][c] == 1` (עד $c = cols - 1$ ו- $r = rows - 1$))) \rightarrow החזרו `true`.

8. סמן ביקרו: `vis[r, c] = true`.

9. נסו ארבעה כיוונים: (`Dfs(r, c-1)` או `Dfs(r, c+1)` או `Dfs(r-1, c)` או `Dfs(r+1, c)`).

10. אם אחת הקראות החזרה `true` \rightarrow החזר `true`; אחרת \rightarrow החזר `false`.

קוד (C#)

```
public static bool FindPath(int[][] grid)
{
    if (grid == null || grid.Length == 0 ||
        grid[0] == null || grid[0].Length == 0)
        return false;

    int rows = grid.Length, cols = grid[0].Length;
```

```

// Start or end blocked → no path
if (grid[0][0] == 1 || grid[rows - 1][cols - 1] == 1)
    return false;

bool[,] vis = new bool[rows, cols];
return Dfs(grid, vis, rows, cols, 0, 0);
}

private static bool Dfs(int[][] grid, bool[,] vis,
                      int rows, int cols, int r, int c)
{
    // Out of bounds
    if (r < 0 || c < 0 || r >= rows || c >= cols)
        return false;

    // Wall or already visited
    if (grid[r][c] == 1 || vis[r, c])
        return false;

    // Reached target
    if (r == rows - 1 && c == cols - 1)
        return true;

    vis[r, c] = true;

    // Explore 4 directions
    return Dfs(grid, vis, rows, cols, r + 1, c) ||
           Dfs(grid, vis, rows, cols, r - 1, c) ||
           Dfs(grid, vis, rows, cols, r, c + 1) ||
           Dfs(grid, vis, rows, cols, r, c - 1);
}

```

MINIMAL

$$\begin{aligned}
 \text{true} &\leftarrow \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix} \bullet \\
 \text{false} &\leftarrow \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \bullet
 \end{aligned}$$

האם קיימת תת-קובוצה שסכוםה שווה ליעד? — SubsetSumExists

דרישות

- משמעותו פעולה (SubsetSumExists(int[] arr, int target) המחזיר true/false).
- לכל איבר במערך יש שתי אפשרויות: לכלול את האיבר בתת-הקובוצה/לא לכלול את האיבר בתת-הקובוצה. השתמשו ברקורסיה לחקירת שתי האפשרויות.

הסבר מלא

נגיד רקורסיבית עם אינדקס: $f(i, t) = f(i + 1, t) \vee f(i + 1, t - a_i)$

כלומר, בכל שלב מחליטים אם להתקדם בלי לבחור את a_i , או לבחור אותו ולהקטין את היעד t בהתאם. מקרי בסיס: $0 = t \leftarrow$ אמת (נמצא תת-קובוצה מתחילה), או $|A| = i \leftarrow$ שקר (הגענו לסוף בלי הצלחה).

דוגמאות

- קלט: $arr = [3, 2, 5]$, $target = 5$.
פלט: true (כי קיימת תת-קובוצה {5} או {3,2} שסכוםה 5).
- קלט: $arr = [1, 4, 6]$, $target = 8$.
פלט: false (אין תת-קובוצה שסכוםה בדיקן 8).
- קלט: $arr = [2, 4, 8]$, $target = 6$.
פלט: true (כי קיימת תת-קובוצה {2,4} שסכוםה 6).

הנחיות לישום (אלגוריתם + קוד)

אלגוריתם

- קלט: מערך arr ויעד t . הגדרו פעולה עזר ($Dfs(i, t)$) שבודקת את הטווח $i \dots |A| - 1$.
- מקרה בסיס $i = 0 \leftarrow t = 0 \leftarrow$ החזירו true (נמצא סכום מדויק).
- מקרה בסיס $i = |A| \leftarrow t < 0 \leftarrow$ החזירו false (אין פתרון במסלול זה).
- מסלול "ללא בחירה" \leftarrow חשבו ($Dfs(i + 1, t)$ ← ממשיכים בלי לכלול את a_i).
- מסלול "עם בחירה" \leftarrow חשבו ($Dfs(i + 1, t - a_i)$ ← ממשיכים לאחר הכללת a_i).
- בדיקה \leftarrow אם אחת משתי הקריאה מהצעדים 4-5 החזירה true; אחרת \leftarrow החזירו false.
- קריאה ראשונית $\leftarrow Dfs(0, target) \leftarrow$ SubsetSumExists(arr, target) קוראת ל- $Dfs(0, target)$.
- סיבוכיות \leftarrow זמן אקספוננציאלי ($O(2^n)$, זיכרון عمוק מחסנית ($O(n)$). (להאצה: זיכרון בין היתר לפי זוג (t, i)).

```

public static bool SubsetSumExists(int[] arr, int target)
{
    return Dfs(arr, 0, target);
}

private static bool Dfs(int[] arr, int i, int t)
{
    if (t == 0) return true;
    if (i == arr.Length || t < 0) return false;
    return Dfs(arr, i + 1, t) || Dfs(arr, i + 1, t - arr[i]);
}

```

שאלות חשיבה

1. מה קורה אם המערך כולל מספרים שליליים — כיצד זה מופיע על האלגוריתם?

שאלה 22

ספירת אלמנטים מאינדקס התחלתי — CountElements

דרישות

- הפעולה `CountElements(int[] arr, int startIndex)` תחזיר את מספר האלמנטים במערך החל מהאינדקס הנתון.
- הימוש יהיה רקורסיבי וללא לולאות.

הסבר מלא

$$C(A, i) = \begin{cases} 0, & i \geq |A| \\ 1 + C(A, i + 1), & \text{תרחא} \end{cases}$$

הנחיות לישום (אלגוריתם + קוד)

אלגוריתם

- קלט: מערך `arr` ואינדקס התחלתי `i = startIndex`.
- מקרה בסיס \leftarrow אם $|A| \geq i \leftarrow$ החזרו 0 (אין עוד אלמנטים לספור).
- צעד רקורסיבי \leftarrow החזרו $1 + C(A, i + 1)$ (סופרים את התא הנוכחי ומתקדמים לאינדקס הבא).
- קריאה ראשונית \leftarrow `CountElements(arr, startIndex)`.

```
public static int CountElements(int[] arr, int startIndex)
{
    if (startIndex >= arr.Length) return 0;
    return 1 + CountElements(arr, startIndex + 1);
}
```

דוגמאות ותנאי קצה

- `CountElements(new int[]{}, 0) → 0`
- `CountElements(new[]{5,7,9}, 1) → 2`

 שאלה 23**תוכנית אינטראקטיבית למספר תנויות באנגלית — CountVowels****דרישות**

- ממשו פעולה רקורסיבית בשם `CountVowels(string str)` שמחזירה את מספר התנויות במחרוזת באנגלית.
- לאחר מכן, כתבו תוכנית קונסול אינטראקטיבית שקוללת מחרוזת מהמשתמש/ת, קוראת `CountVowels`, ומדפיסה את מספר התנויות.
- התנויות הן `a, e, i, o, u`.

הסבר מלא

ברקורסיה נבדוק את התו הראשון, ונוסיף 1 אם הוא תנועה; נתקדם על שאר המחרוזת עד שהמחרוזת ריקה.

$$V(s) = (s = \epsilon ? 0 : [s[0] \in \{a, e, i, o, u, A, E, I, O, U\}] + V(s[1..]))$$

הנחיות ליישום (אלגוריתם + קוד)**אלגוריתם**

1. אם המחרוזת ריקה ← החזרו 0.
2. בדקו אם `s[0]` שייך/ת לקבוצת התנויות.
3. החזרו (0 או 1) + קראת רקורסיבית `CountVowels` על `[1..s]`.

```

public static int CountVowels(string str) {
    if (str is null) throw new ArgumentNullException(nameof(str));
    if (str.Length == 0) return 0;
    bool isVowel = "aeiouAEIOU".IndexOf(str[0]) >= 0;
    return (isVowel ? 1 : 0) + CountVowels(str.Substring(1));
}

static void Main() {
    Console.Write("Enter an English string: ");
    string s = Console.ReadLine() ?? "";
    Console.WriteLine(CountVowels(s));
}

```

שאלות הבנה וחשיבה

1. כיצד תתמכנו בקבוצת תנויות שונה (לשפות אחרות)?
2. איך ניתן לצמצם ייצירת מחרוזות ביןיהם (ללא `Substring`)?

דוגמאות ותנאי קצה

- קלט: "Hello" → פלט: 2
- קלט: מחרוזת ריקה → פלט: 0

 שאלה 24**הdfs 1..n — הדפסת עולה (רקורסיבי, void)****דרישות**

- הפעולה (`PrintReverse`) מקבלת מספר שלם חיובי אחד.
- עליה להדפיס את המספרים מ-1 עד `n` בסדר עולה, כאשר כל מספר מופרד ברווח.
- הפעולה חייבת להיות רקורסיבית ואני מחייבת ערך (`void`).
- אם `0 <= n` — אין להדפיס דבר.

הסבר מלא

נשתמש בהדפסה לאחר הקראיה הרקורסיבית: תחילת נדפיס את התחום `1..n`, ואז נדפיס את `n`. למרות שם הפעולה, סדר ההדפסה עולה.

הנחיות ליישום (אלגוריתם + קוד)

1. אם $n \leq 0$ ← עצירה.

2. קראו רקורסיבית ל-($n-1$).PrintReverse($n-1$)

3. הדפיסו את n ואחריו רווח.

קוד (C#)

```
public static void PrintReverse(int n) {  
    if (n <= 0) return;  
    PrintReverse(n - 1);  
    Console.Write(n + " ");  
}
```

דוגמא

5 4 3 2 1 → PrintReverse(5) •