**שפות תכנות מונחי עצמים**

**תרגיל מספר 1**

עליך לממש מחלקה בשם Number המייצגת מספר בינארי והתומכת באופרטור הסכום "+" והכפל "\*".

הייצוג הפנימי של המספר יהיה מערך של שלמים בגודל n בשם digits אשר כל כניסה תכיל את הערך 0 או 1.

לדעתי הכי נוח ש-digits[0] יכיל את הספרה המשמעותית ביותר ו-digits[n-1] את המשמעותית הכי פחות.

לדוגמא, אם n=8 המספר הוא 11001 (= 25 בדצימל) הייצוג הפנימי יהיה

digits[0] =0, digits[1] =0, digits[2] =0, digits[3] =1, digits[4] =1,

digits[5] =0, digits[6] =0, digits[7] =1

הגודל n הוא של המערך digits תמיד חזקה של 2 הוא בלתי תלוי בערך עצמו ונקבע ברגע ההצבה (בנאי או השמה) המינימום שנחוץ כאשר זה משתמע.

האלגוריתמים למימוש יהיו אלו שלמדתם בבית ספר יסודי:

1011110

+

1100011

-----------

11000001

ואילו כפל

1011110

\*

1100011

-----------

1011110

1011110

1011110

1011110

----------------------

10011110011010

(עוברים על הביטים של המספר השני ף כל פעם עושים shift למספר הראשון ומסכמים **אם** הביט במספר השני שווה ל-1).

המתודות שעל Number לתמוך בהם הינם:

1. מאפיין N המאפשר לקוד לקוח לקרוא אבל לא לשנות את גודל המערך n.
2. בנאי

public Number(String value, int n)

המאפשר הקצאת מופע מחלקה עם גודל n של מערך ה-digits וערך התחלתי של המערך נקבע ע"י המחרוזת String. אם יש צורך ממירים את n לחזקה הקרובה של 2 מלמעלה (למשל 24 יומר ל-32) דוגמא של קריאה יכולה להיות

n1 = new Number("110001101", 32);

1. מתודה

ToString();

היוצרת מחרוזת המייצגת את הערך של המופע בדיוק בגודל הנחוץ (לאו דווקא תלוי ב-n).

1. העמסת אופרטור הסכום "+". (ההנחה שגודל n של המערך digits של 2 המחוברים שווה ומספיק גדול להכיל את התוצאה).
2. העמסת אופרטור הכפל "\*". (ההנחה שגודל n של המערך digits של 2 הגורמים שווה ומספיק גדול להכיל את התוצאה).
3. העמסת אופרטור הצבה "=". במקרה של השמה

Number n1 = “10011”;

הגודל n של המערך digits יהיה החזקה הראשונה של 2 שהוא לפחות פי 2 מאורך המחרוזת.

1. בנאי ללא פרמטרים היוצר מספר ריק.
2. מאפיין להשמה ולקריאה HexString, ייצוג נמספר בהקסה דצימל, לדוגמא

Number n1;

n1 = new Number();

n1.HexString = "F4215";

Console.WriteLine("{0}",

n1.HexString);

1. אינדקסר לקריאה בלבד, למשל

Console.WriteLine("digits n1:");

for(i=0; i < n1.N; i++)

Console.Write(" {0} ", n1[i]);

Console.WriteLine();

לדוגמא, הפלט של התוכנית הבאה:

class MainClass

{

public static void Main(String [] args)

{

Number n1, n2, n3;

int i;

n1 = new Number();

n2 = new Number();

n1.HexString = "F4215";

n2.HexString = "A352B";

n3 = n1 + n2;

Console.WriteLine("n1.N = " + n1.N);

Console.WriteLine("n2.N = " + n2.N);

Console.WriteLine("n3.N = " + n3.N);

Console.WriteLine("{0} + {1} = {2}",

n1.ToString(), n2.ToString(), n3.ToString());

Console.WriteLine("{0} + {1} = {2}",

n1.HexString, n2.HexString, n3.HexString);

Console.WriteLine("digits n1:");

for(i=0; i < n1.N; i++)

Console.Write(" {0} ", n1[i]);

Console.WriteLine();

n3 = n1 \* n2;

Console.WriteLine("n1.N = " + n1.N);

Console.WriteLine("n2.N = " + n2.N);

Console.WriteLine("n3.N = " + n3.N);

Console.WriteLine("{0} \* {1} = {2}",

n1.ToString(), n2.ToString(), n3.ToString());

Console.WriteLine("{0} \* {1} = {2}",

n1.HexString, n2.HexString, n3.HexString);

Console.WriteLine("n1.N = " + n1.N);

n1 = "1011110";

n2 = "1100011";

n3 = n1 + n2;

Console.WriteLine("n1.N = " + n1.N);

Console.WriteLine("n2.N = " + n2.N);

Console.WriteLine("n3.N = " + n3.N);

Console.WriteLine("{0} + {1} = {2}",

n1.ToString(), n2.ToString(), n3.ToString());

Console.WriteLine("{0} + {1} = {2}",

n1.HexString, n2.HexString, n3.HexString);

n3 = n1 \* n2;

Console.WriteLine("n1.N = " + n1.N);

Console.WriteLine("n2.N = " + n2.N);

Console.WriteLine("n3.N = " + n3.N);

Console.WriteLine("{0} \* {1} = {2}",

n1.ToString(), n2.ToString(), n3.ToString());

Console.WriteLine("{0} \* {1} = {2}",

n1.HexString, n2.HexString, n3.HexString);

Console.WriteLine("n1.N = " + n1.N);

n1 =

new

Number("1100011010101110011111", 128);

n2 =

new

Number("1011101001111000001111", 128);

Console.WriteLine("n1.N = " + n1.N);

Console.WriteLine("n2.N = " + n2.N);

Console.WriteLine("n3.N = " + n3.N);

n3 = n1 + n2;

Console.WriteLine("{0} + {1} = {2}",

n1.ToString(), n2.ToString(), n3.ToString());

Console.WriteLine("{0} + {1} = {2}",

n1.HexString, n2.HexString, n3.HexString);

n3 = n1 \* n2;

Console.WriteLine("n1.N = " + n1.N);

Console.WriteLine("n2.N = " + n2.N);

Console.WriteLine("n3.N = " + n3.N);

Console.WriteLine("{0} \* {1} = {2}",

n1.ToString(), n2.ToString(), n3.ToString());

Console.WriteLine("{0} \* {1} = {2}",

n1.HexString, n2.HexString, n3.HexString);

n1 =

new

Number("11000110101011100111110001111111001111001111100111100110011", 128);

n2 =

new

Number("10111010011110000011111111111100000111000111000011100011111", 128);

n3 = n1 + n2;

Console.WriteLine("n1.N = " + n1.N);

Console.WriteLine("n2.N = " + n2.N);

Console.WriteLine("n3.N = " + n3.N);

Console.WriteLine("{0} + {1} = {2}",

n1.ToString(), n2.ToString(), n3.ToString());

Console.WriteLine("{0} + {1} = {2}",

n1.HexString, n2.HexString, n3.HexString);

n3 = n1 \* n2;

Console.WriteLine("n1.N = " + n1.N);

Console.WriteLine("n2.N = " + n2.N);

Console.WriteLine("n3.N = " + n3.N);

Console.WriteLine("{0} \* {1} = {2}",

n1.ToString(), n2.ToString(), n3.ToString());

Console.WriteLine("{0} \* {1} = {2}",

n1.HexString, n2.HexString, n3.HexString);

} // Main

} // MainClass

יהיה:

n1.N = 32

n2.N = 32

n3.N = 32

11110100001000010101 + 10100011010100101011 = 110010111011101000000

F4215 + A352B = 197740

digits n1:

0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 1 0 0 0 0 1 0 0 0 0 1 0 1 0 1

n1.N = 32

n2.N = 32

n3.N = 64

11110100001000010101 \* 10100011010100101011 = 1001101111000000000100000111001010000111

F4215 \* A352B = 9BC0107287

n1.N = 32

n1.N = 8

n2.N = 8

n3.N = 8

1011110 + 1100011 = 11000001

5E + 63 = C1

n1.N = 8

n2.N = 8

n3.N = 16

1011110 \* 1100011 = 10010001011010

5E \* 63 = 245A

n1.N = 8

n1.N = 128

n2.N = 128

n3.N = 16

1100011010101110011111 + 1011101001111000001111 = 11000000100100110101110

31AB9F + 2E9E0F = 6049AE

n1.N = 128

n2.N = 128

n3.N = 256

1100011010101110011111 \* 1011101001111000001111 = 10010000101110000001011001110011000001010001

31AB9F \* 2E9E0F = 90B81673051

n1.N = 128

n2.N = 128

n3.N = 128

11000110101011100111110001111111001111001111100111100110011 + 10111010011110000011111111111100000111000111000011100011111 = 110000001001001101011110001111011010110010110101011001010010

63573E3F9E7CF33 + 5D3C1FFE0E3871F = C0935E3DACB5652

n1.N = 128

n2.N = 128

n3.N = 256

11000110101011100111110001111111001111001111100111100110011 \* 10111010011110000011111111111100000111000111000011100011111 = 1001000010111000000110011110011101101000011010101011011010010110001101100001100000101011011011100011101111110000101101

63573E3F9E7CF33 \* 5D3C1FFE0E3871F = 242E0679DA1AADA58D860ADB8EFC2D