2. معطى توجيهان إلى سلسلتَي حلقات:

numHead – توجيه إلى سلسلة حلقات لأعداد من نمط صحيح.

charHead – توجيه إلى سلسلة حلقات من نمط رمز (char)، تُمثَّل فيها العمليّتان الحسابيّتان الجمع والطرح ('+' ، '−').

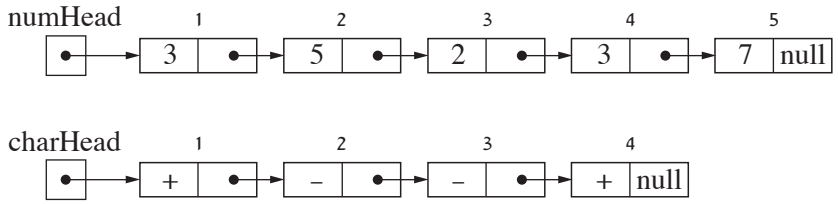
ملاحظة: كمّيّة الحلقات في سلسلة الأعداد هي أكبر بـ 1 من كمّيّة الحلقات في سلسلة الرموز.

الدمج بين السلسلتَيْن يُمثِّل تسلسل عمليّات حسابيّة، كما هو مفصَّل أمامكم:

الرمز في المكان 1 في سلسلة الرموز يصف العمليّة التي تأتي قبل العدد الذي في المكان 2 في سلسلة الأعداد، والرمز في المكان 2 يصف العمليّة التي تأتي قبل العدد الذي في المكان 3، والرمز في المكان 3 يصف العمليّة التي تأتي قبل العدد الذي في المكان 4، وهكذا حتّى نهاية السلسلتَيْن.

انتبهوا: لا توجد عمليّة حسابيّة قبل العدد الأوّل في سلسلة الأعداد.

مثال: السلسلتان اللتان أمامكم تُمثِّلان تسلسل العمليّات الحسابيّة: $7+3-2-5+3$.



أ. طبِّقوا العمليّة التي أمامكم:

Java – public static int eval (Node<Integer> numHead, Node<Character> charHead, int len)

C# – public static int Eval (Node<int> numHead, Node<char> charHead, int len)

len هو كمّيّة **الأعداد** التي نرغب في حسابها من داخل سلسلة الأعداد (لا توجد عمليّة حسابيّة قبل العدد الأوّل، لذلك كمّيّة الرموز هي len–1 بالتلاؤم).

تُعيد العمليّة نتيجة حساب len أعداد بتسلسل، بدءًا من بداية سلسلة الأعداد وبداية سلسلة الرموز.

افترِضوا أنّ len هو أكبر من 0 . لا تُغيِّروا السلسلتَيْن المتلقّاتَيْن.

ملاحظة: إذا كان len أكبر من كمّيّة الحلقات في سلسلة الأعداد، تُعيد العمليّة نتيجة حساب جميع الأعداد في السلسلة.

مثال: بالنسبة للسلسلتَيْن في المثال أعلاه وَ $\mathbf{len = 1}$ ، تُعيد العمليّة 3 ، لأنّه هو العدد الأوّل في سلسلة الأعداد (لِكَوْنه هو العدد الأوّل والوحيد، لا توجد أيّة عمليّة حسابيّة يجب تنفيذها).

مثال إضافيّ: بالنسبة لنفس السلسلتَيْن وَ $\mathbf{len = 3}$ ، تُعيد العمليّة 6 $(3+5-2=6)$.

مثال إضافيّ: بالنسبة لنفس السلسلتَيْن وَ $\mathbf{len = 8}$ ، تُعيد العمليّة 10 $(3+5-2-3+7=10)$.

الشرح: عدد الحلقات في سلسلة الأعداد هو أقلّ من 8 ، لذلك تُعيد العمليّة نتيجة حساب جميع الأعداد في السلسلة.

(انتبهوا: تكملة السؤال في الصفحة التالية.)

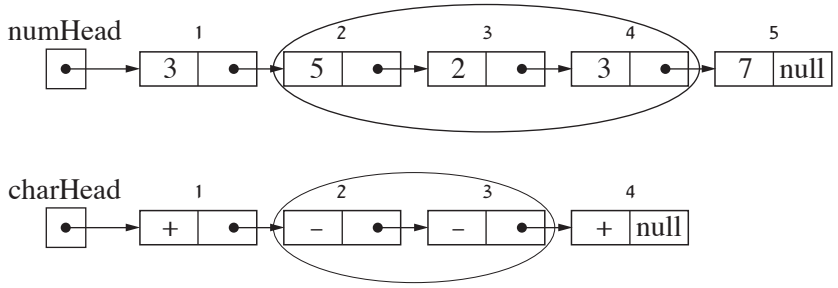
ب. طبِّقوا العمليّة التي أمامكم:

Java – public static boolean match (Node<Integer> numHead, Node<Character> charHead, int len, int val)

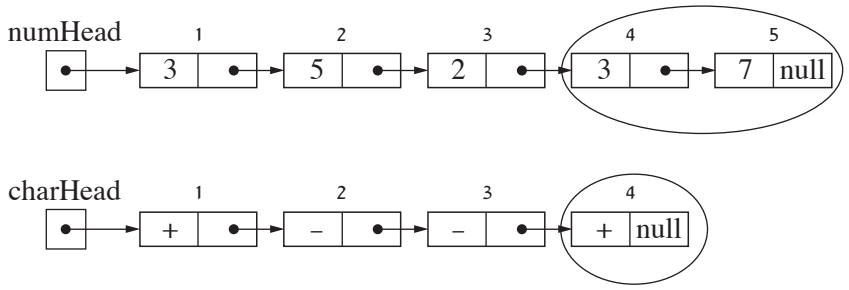
C# – public static bool Match (Node<int> numHead, Node<char> charHead, int len, int val)

تُعيد العمليّة true إذا كانت هناك حلقة **في مكان ما** في سلسلة الأعداد وحلقة **في مكان مطابق** في سلسلة الرموز، اللذين منهما فصاعدًا يوجد **len** أعداد بتسلسل نتيجةُ حسابِها هي **val** (أي أنّه إذا بدأنا الحساب **من الحلقة الثانية** في سلسلة الأعداد، ستكون العمليّة الحسابيّة الأولى أيضًا هي **الحلقة الثانية** في سلسلة الرموز). خلاف ذلك، تُعيد العمليّة false.
بإمكانكم استعمال العمليّة التي كتبتموها في البند "أ". افترِضوا أنّ len أكبر من 0. لا تُغيِّروا السلسلتَيْن المتلقّاتَيْن.
انتبهوا: لا توجد عمليّة حسابيّة قبل العدد الأوّل في الحساب (عدد الرموز المطلوب هو len–1).
ملاحظة: إذا كان أقلّ من len أعداد من حلقة معيّنة حتّى نهاية السلسلة – يتمّ الحساب حتّى نهاية السلسلة.

مثال: بالنسبة للسلسلتَيْن المعروضتَيْن في المثال أعلاه، وَ **len = 3** وَ **val = 0**، تُعيد العمليّة **true**، لأنّه بدءًا **من الحلقة الثانية** في سلسلة الأعداد **ومن الحلقة الثانية** في سلسلة الرموز، توجد 3 أعداد بتسلسل نتيجةُ حسابِها هي 0 (5 – 2 – 3 = 0).



مثال إضافيّ: بالنسبة للسلسلتَيْن المعروضتَيْن في المثال أعلاه، وَ **len = 3** وَ **val = 10**، تُعيد العمليّة **true**، لأنّه بدءًا **من الحلقة الرابعة** في سلسلة الأعداد **ومن الحلقة الرابعة** في سلسلة الرموز وحتّى نهاية سلسلة الأعداد، نتيجة الحساب هي 10 (3 + 7 = 10).



/ يتبع في صفحة 6 /