Write a program to add two numbers in base 14

Asked by Anshya.

Below are the different ways to add base 14 numbers.

Method 1
Thanks to Raj for suggesting this method.

```
1. Convert both i/p base 14 numbers to base 10.
2. Add numbers.
3. Convert the result back to base 14.
```

Method 2
Just add the numbers in base 14 in same way we add in base 10. Add numerals of both numbers one by one from right to left. If there is a carry while adding two numerals, consider the carry for adding next numerals.

Let us consider the presentation of base 14 numbers same as hexadecimal numbers

```
A --> 10
B --> 11
C --> 12
D --> 13
```

```
Example:
   num1 =       1  2  A
   num2 =       C  D  3

   1. Add A and 3, we get 13(D). Since 13 is smaller than
14, carry becomes 0 and resultant numeral becomes D

  2. Add 2, D and carry(0). we get 15. Since 15 is greater
than 13, carry becomes 1 and resultant numeral is 15 - 14 = 1

  3. Add 1, C and carry(1). we get 14. Since 14 is greater
than 13, carry becomes 1 and resultant numeral is 14 - 14 = 0

Finally, there is a carry, so 1 is added as leftmost numeral and the result becomes
101D
```

Implementation of Method 2

```
# include <stdio.h>
```

```
# include <stdlib.h>
# define bool int

int getNumeralValue(char );
char getNumeral(int );

/* Function to add two numbers in base 14 */
char *sumBase14(char *num1,  char *num2)
{
   int l1 = strlen(num1);
   int l2 = strlen(num2);
   char *res;
   int i;
   int nml1, nml2, res_nml;
   bool carry = 0;

   if(l1 != l2)
   {
     printf("Function doesn't support numbers of differe
            " lengths. If you want to add such numbers t
            " prefix smaller number with required no. of
     getchar();
     assert(0);
   }

   /* Note the size of the allocated memory is one
     more than i/p lenghts for the cases where we
     have carry at the last like adding D1 and A1 */
   res = (char *)malloc(sizeof(char)*(l1 + 1));

   /* Add all numerals from right to left */
   for(i = l1-1; i >= 0; i--)
   {
     /* Get decimal values of the numerals of
       i/p numbers*/
     nml1 = getNumeralValue(num1[i]);
     nml2 = getNumeralValue(num2[i]);

     /* Add decimal values of numerals and carry */
     res_nml = carry + nml1 + nml2;

     /* Check if we have carry for next addition
        of numerals */
     if(res_nml >= 14)
     {
       carry = 1;
       res_nml -= 14;
     }
     else
     {
       carry = 0;
     }
     res[i+1] = getNumeral(res_nml);
   }

   /* if there is no carry after last iteration
      then result should not include 0th character
      of the resultant string */
```

```
    if(carry == 0)
      return (res + 1);

    /* if we have carry after last iteration then
      result should include 0th character */
    res[0] = '1';
    return res;
}

/* Function to get value of a numeral
  For example it returns 10 for input 'A'
  1 for '1', etc */
int getNumeralValue(char num)
{
  if( num >= '0' && num <= '9')
    return (num - '0');
  if( num >= 'A' && num <= 'D')
    return (num - 'A' + 10);

  /* If we reach this line caller is giving
    invalid character so we assert and fail*/
  assert(0);
}

/* Function to get numeral for a value.
  For example it returns 'A' for input 10
  '1' for 1, etc */
char getNumeral(int val)
{
  if( val >= 0 && val <= 9)
    return (val + '0');
  if( val >= 10 && val <= 14)
    return (val + 'A' - 10);

  /* If we reach this line caller is giving
    invalid no. so we assert and fail*/
  assert(0);
}

/*Driver program to test above functions*/
int main()
{
    char *num1 = "DC2";
    char *num2 = "0A3";

    printf("Result is %s", sumBase14(num1, num2));
    getchar();
    return 0;
}
```

Notes:

Above approach can be used to add numbers in any base. We don’t have to do string operations if base is smaller than 10.