```
/*
Prolog statements to calculate the sum of individual digits
composing a positive, integer-valued argument. In addition,
a list of the character-based representation of these digits
is returned in sorted order.

Michael E. Sparks, 8 Nov 2020

?- sum_digits(629174853,ListOfSorted,SumOfDigits).
ListOfSorted = ['1', '2', '3', '4', '5', '6', '7', '8', '9'],
SumOfDigits = 45.

?- X is 9 * 10 / 2.
X = 45.

?- number_string(1234,X), string_chars(X,[H|T]), number_string(Y,[H]), Z is Y +
1.
X = "1234",
H = '1',
T = ['2', '3', '4'],
Y = 1,
Z = 2.
*/

sum_digits(Num,SortedDigits,Sum) :-
    number_string(Num,NumAsString),
    string_chars(NumAsString,DigitsAsChars),
    quicksort(DigitsAsChars,SortedDigits), !,
    sum_digital_rep_of_chars(SortedDigits,Sum).

% As an independent exercise, contemplate how to rewrite this
% predicate in a manner that places constant pressure on the stack.
sum_digital_rep_of_chars([],0).

sum_digital_rep_of_chars([Hc|T],Sum) :-
    number_string(Hi,[Hc]),
    sum_digital_rep_of_chars(T,Sum1), % ignoring tail-call optimization! :(
    Sum is Hi + Sum1.

/* As a bonus, here's a specification in Haskell
   of Sir Tony Hoare's masterpiece:

quicksort :: (Ord a) => [a] -> [a]
quicksort [] = []
quicksort (x:xs) =
  let smaller = filter (<=x) xs
      larger = filter (>x) xs
  in quicksort smaller ++ [x] ++ quicksort larger

*/
quicksort([],[]).

quicksort([Pivot|Tail],Sorted) :-
    partition(Pivot,Tail,Smaller,Larger),
    quicksort(Smaller,SortedSmaller),
    quicksort(Larger,SortedLarger),
    append(SortedSmaller,[Pivot|SortedLarger],Sorted).

partition(_,[],[],[]).
```

```
partition(Pivot,[X|T],[X|Smaller],Larger) :-
    X @=< Pivot, !,
    partition(Pivot,T,Smaller,Larger).

partition(Pivot,[X|T],Smaller,[X|Larger]) :-
    X @> Pivot, !, % the .GT. check's technically unnecessary
    partition(Pivot,T,Smaller,Larger).
```