6 Programming in C and C++ (djg11)

Note: Part (*a*) of this question should be answered **without any reference to C++**.

(*a*) The C language has hardly any native support for character strings, with facilities mainly being provided by standard libraries.

(*i*) What language features or behaviours *must* a C compiler implement with respect to strings? [2 marks]

(*ii*) When might it be helpful or harmful to make sure every string is held at most once in memory? [4 marks]

(*iii*) In a programming exercise you are asked to write a C routine that removes the filename suffix, i.e. the last dot and any chars after it. It must be implemented entirely as native C code, not using `malloc` or any libraries, for use cases such as

```
const char *sans_suffix = remove_suffix("foobar.txt");
```

Is this possible? Give justification for your answer, discussing the possibilities for using the stack or other dynamic allocation or static storage. Code is not necessary but could be included to explain your answer. [4 marks]

(*b*) For this part of the question, consider string processing in C++.

(*i*) Does C++ provide any additional native support for string processing with respect to C? [1 mark]

(*ii*) Assuming `std::string` is not available, a C++ class is needed that provides convenient RAII memory management for strings. This will automatically deallocate string memory (without relying on a garbage collector). Discuss whether it is possible in this scheme to ensure that each string is held only once. [3 marks]

(*iii*) Using `new char [n]` and `delete`, but without using any library functions, sketch C++ code that implements Part (*b*)(*ii*), with strings potentially stored more than once. Include an overload of the `+=` operator which extends a string with another one. Add comments or arrows or proof arguments that demonstrate that no deleted memory is read. [6 marks]