CS100 Spring 2024

Quiz 1

Apr 24, 2024

1. (15 points) Your name: _____. Your student ID: _____.

   Your email: _____@shanghaitech.edu.cn

2. (40 points) (C) The following functions are all related to "removing digits from a string". For each function, does it correctly implement the behavior described in its documentation (the comments before it)? If not, what is wrong with it?

   (a)
   ```
   /// @brief Writes a string to 'dest' obtained from 'source' by removing all the digits.
   /// @param dest Points to a block of memory large enough to hold the result.
   /// @param source Points to a null-terminated byte string.
   void remove_digits(char *dest, const char *source) {
     while (*source != '\0') {
       if (!isdigit(*source))
         *dest++ = *source;
       ++source;
     }
   }
   ```

   Solution: Incorrect. It forgets to put a `'\0'` at the end.

   (b)
   ```
   /// @brief Writes a string to 'dest' obtained from 'source' by removing all the digits.
   /// @param dest Points to a block of memory large enough to hold the result.
   /// @param source Points to a null-terminated byte string.
   void remove_digits(char *dest, const char *source) {
     strcpy(dest, source);
     for (char *i = dest; *i != '\0';) {
       if (isdigit(*i)) {
         for (char *j = i; *j != '\0'; ++j)
           *j = *(j + 1);
       } else
         ++i;
     }
   }
   ```

   Solution: Incorrect. The memory block pointed to by `dest` is only required to be large enough to hold the result. But the implementation actually requires more, since it copies the entire string `source` to it in the beginning.

(c)
```
/// @brief Writes a string to 'dest' obtained from 'source' by removing all the digits.
/// @param dest Points to a block of memory large enough to hold the result.
/// @param source Points to a null-terminated byte string.
void remove_digits(char *dest, const char *source) {
  int cnt = 0;
  for (const char *i = source; *i != '\0'; ++i)
    if (!isdigit(*i))
      ++cnt;
  dest = calloc(cnt + 1, 1);
  for (const char *i = source; *i != '\0'; ++i)
    if (!isdigit(*i))
      *dest++ = *i;
}
```

Solution: Incorrect. `dest` already points to a block of memory, and the function should write the result there, instead of allocating memory itself.

(d)
```
/// @brief Removes the digits in the given string.
/// @param str Points to a null-terminated byte string.
/// @return int The number of digits removed.
int remove_digits(char *str) {
  int digit_cnt = 0;
  for (char *i = str; *i != '\0'; ++i) {
    if (isdigit(*i)) {
      ++digit_cnt;
      for (char *j = i; *j != '\0'; ++j)
        *j = *(j + 1);
    }
  }
  return digit_cnt;
}
```

Solution: Incorrect. When two consecutive digits appear, the second will be skipped.

3. (15 points) (C++) Define a function `length_sum` that computes the sum of the length of all strings in a `std::vector<std::string>`. Fill in the blanks below.

```
#include <vector>
#include <string>
#include <cassert>

std::size_t length_sum(/* (a) */) {
  std::size_t sum = 0;
  for (/* (b) Use a range-based for loop. */)
    sum += /* (c) */;
  return sum;
}
int main() {
  std::vector<std::string> vs{"hello", "C++", "world"};
  assert(length_sum(vs) == 13); // This assertion should succeed.
}
```

(a) `const std::vector<std::string> &strings`

(b) `const std::string &s : strings`

Also correct: `const auto &s : strings`
Also correct: `auto &s : strings`

(c) `s.size()` or `s.length()` or `std::size(s)`

Also correct: `size(s)` (due to ADL)