```h
#ifndef STRINGBUILDER_H
#define STRINGBUILDER_H

typedef struct stringbuilder_tag {
    char* cstr;             /* Must be first member in the struct! */
    int   pos;
    int   size;
    int   reallocs;         /* Performance metric to record the number of string reallocations */
} stringbuilder;

/**
* Creates a new stringbuilder with the default chunk size
*
*/
stringbuilder* sb_new();

/**
* Destroys the given stringbuilder.  Pass 1 to free_string if the underlying c string should also be freed
*/
void sb_destroy(stringbuilder* sb, int free_string);

/**
* Creates a new stringbuilder with initial size at least the given size
*/
stringbuilder* sb_new_with_size(int size);

/**
* Resets the stringbuilder to empty
*/
void sb_reset(stringbuilder* sb);

/**
* Appends the given character to the string builder
*/
void sb_append_ch(stringbuilder* sb, const char ch);

/**
* Appends at most length of the given src string to the string buffer
*/
void sb_append_strn(stringbuilder* sb, const char* src, int length);

/**
* Appends the given src string to the string builder
*/
void sb_append_str(stringbuilder* sb, const char* src);

/**
* Allocates and copies a new cstring based on the current stringbuilder contents
*/
char* sb_make_cstring(stringbuilder* sb);
```

```
51
52  /**
53  * Returns the stringbuilder as a regular C String
54  */
55  #define sb_cstring(sb) ((sb)->cstr)
56
57  #endif // STRINGBUILDER_H
58
```