```c
//
// Created by hfwei on 2023/12/19.
//
// sds.h: https://github.com/huangz1990/redis-3.0-annotated/blob/unstable/src/sds.h
// sds.c: https://github.com/huangz1990/redis-3.0-annotated/blob/unstable/src/sds.c
//

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <assert.h>

typedef char *sds;

struct sdshdr {
  int len;
  int free;
  char buf[];
};

static inline size_t sdslen(const sds s) {
  struct sdshdr *sh = (void *) (s - sizeof(struct sdshdr));
  return sh->len;
}

static inline size_t sdsavail(const sds s) {
  struct sdshdr *sh = (void *) (s - sizeof(struct sdshdr));
  return sh->free;
}

sds sdsnewlen(const void *init, size_t initlen);
// sds sdsnew(const char *init);

void sdsfree(sds s);

sds sdsMakeRoomFor(sds s, size_t addlen);
sds sdscatlen(sds s, const void *t, size_t len);
sds sdscpylen(sds s, const char *t, size_t len);

int main(void) {
  struct sdshdr *sh;

  sds x = sdsnewlen("foo", 3);
  assert(sdslen(x) == 3);

  // adding test-case for sdscatlen
  x = sdscatlen(x, "bar", 3);
  assert(sdslen(x) == 6);
  assert(strcmp(x, "foobar") == 0);

  // adding test-case for sdscpylen
```

```c
  x = sdscpylen(x, "a", 1);
  assert(sdslen(x) == 1);
  assert(strcmp(x, "a") == 0);

  return 0;
}

sds sdsnewlen(const void *init, size_t initlen) {
  struct sdshdr *sh;

  sh = malloc(sizeof(struct sdshdr) + initlen + 1);
  if (sh == NULL) {
    return NULL;
  }

  sh->len = initlen;
  sh->free = 0;

  if (initlen && init) {
    memcpy(sh->buf, init, initlen);
  }

  sh->buf[initlen] = '\0';

  return (char *) sh->buf;
}

void sdsfree(sds s) {
  if (s == NULL) {
    return;
  }

  free(s - sizeof(struct sdshdr));
}

sds sdsMakeRoomFor(sds s, size_t addlen) {
  struct sdshdr *sh, *newsh;
  size_t free = sdsavail(s);
  size_t len, newlen;

  if (free >= addlen) {
    return s;
  }

  len = sdslen(s);
  sh = (void *) (s - sizeof(struct sdshdr));
  newlen = (len + addlen) * 2;
  newsh = realloc(sh, sizeof(struct sdshdr) + newlen + 1);
  if (newsh == NULL) {
    return NULL;
  }

  newsh->free = newlen - len;
```

```
105   return newsh->buf;
106 }
107
108 sds sdscatlen(sds s, const void *t, size_t len) {
109   struct sdshdr *sh;
110   size_t curlen = sdslen(s);
111
112   s = sdsMakeRoomFor(s, len);
113   if (s == NULL) {
114     return NULL;
115   }
116
117   sh = (void *) (s - sizeof(struct sdshdr));
118   memcpy(s + curlen, t, len);
119   sh->len = curlen + len;
120   sh->free = sh->free - len;
121   s[curlen + len] = '\0';
122
123   return s;
124 }
125
126 sds sdscpylen(sds s, const char *t, size_t len) {
127   struct sdshdr *sh = (void *) (s - sizeof(struct sdshdr));
128   size_t totlen = sh->free + sh->len;
129
130   if (totlen < len) {
131     s = sdsMakeRoomFor(s, len - sh->len);
132     if (s == NULL) {
133       return NULL;
134     }
135     sh = (void *) (s - sizeof(struct sdshdr));
136     totlen = sh->free + sh->len;
137   }
138
139   memcpy(s, t, len);
140   s[len] = '\0';
141   sh->len = len;
142   sh->free = totlen - len;
143
144   return s;
145 }
```