```
//
// Solution to C dynamic strings allocation and sorting primer by
// Golden G. Richard III (@nolaforensix).
//

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

#define BUFSIZE 1024

/*
int cmp(const void *p1, const void *p2) {
  char *s1 = *(char **)p1;
  char *s2 = *(char **)p2;
  return strcmp(s1, s2);
}
*/

int cmp(const void *p1, const void *p2) {
  return strcmp( *(char **)p1, *(char **)p2);
}


int main(int argc, char *argv[]) {
  char **strings;
  int i,n, len;
  char buf[BUFSIZE+1];

  printf("Enter # of strings:");
  fflush(stdout);
  fgets(buf, BUFSIZE, stdin);
  n=atoi(buf);
  strings=malloc(n * sizeof(char *)); // <---
  for (i=0; i < n; i++) {
    printf("Enter string # %d.\n", i+1);
    fgets(buf, BUFSIZE-1, stdin);
    len=strlen(buf);
    if (buf[len-1] == '\n') {
      buf[len-1]=0;
    }
    strings[i]=malloc((strlen(buf)+1) * sizeof(char));
    strcpy(strings[i], buf);
  }

  qsort(strings, n, sizeof(char *), cmp);

  printf("Sorted strings:\n");
  for (i=0; i < n; i++) {
    printf("%s\n", strings[i]);
    free(strings[i]);   // get used to doing this--free() the memory
                        // when it is no longer needed.  But beware
                        // use-after-free errors--do not use the
                        // memory again once you free it!
  }

  free(strings);        // as before.  Clean up memory when it's no
                        // longer needed.
}
```