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

struct pila {
    char *cmd;
    struct pila *next;
};

struct pila* search_prefix(struct pila *history, char *target)
{
    int l = strlen(target);
    while(history) {
        if(strncmp(history->cmd, target, l) == 0) return history;
        history = history->next;
    }
    return NULL;
}

struct pila* search_cmd(struct pila *history, char *target)
{
    while(history) {
        if(strcmp(history->cmd, target) == 0) return history;
        history = history->next;
    }
    return NULL;
```

```
 27  }
 28
 29  struct pila* push(struct pila *history, char *buffer)
 30  {
 31      struct pila *new = malloc(sizeof(struct pila));
 32      if(new) {
 33          new->next = history;
 34          new->cmd = strdup(buffer);
 35          if(new->cmd) {
 36              return new;
 37          } else {
 38              // non uso new
 39              free(new);
 40              return history;
 41          }
 42      }
 43      return history;
 44  }
 45
 46  void free_node(struct pila * pt)
 47  {
 48      free(pt->cmd);
 49      free(pt);
 50  }
 51
 52
```

```
53  void delete_node(struct pila *history, struct pila *target)
54  {
55      while(history->next) {
56          if(history->next == target) {
57              history->next = target->next;
58              return;
59          }
60          history = history->next;
61      }
62  }
63
64  int main(int argc, char *argv[])
65  {
66      FILE *f = stdin, *fout=NULL;
67      int i;
68
69      for(i=1; i<argc; i++) {
70          if(strncmp(argv[i], "if=", 3) == 0) {
71              f = fopen(argv[i]+3, "r");
72              if(!f) {
73                  perror(argv[i]);
74                  exit(1);
75              }
76              continue;
77          }
78          if(strncmp(argv[i], "of=", 3) == 0) {
```

```
 79              fout = fopen(argv[i]+3, "w");
 80              if(!fout) {
 81                  perror(argv[i]);
 82                  exit(1);
 83              }
 84              continue;
 85          }
 86          if(strcmp(argv[i], "--help") == 0) {
 87              printf("uso: %s [--help] [in=file]\n", argv[0]);
 88              return 0;
 89          }
 90          fprintf(stderr, "parametro non gestito: %s\n", argv[i]);
 91      }
 92      char buffer[1000];
 93      struct pila *history = NULL;
 94      while(fgets(buffer, sizeof(buffer), f)) {
 95          int l = strlen(buffer);
 96          buffer[l-1] = 0; // elimino il carattere di newline
 97          // non vengono gestite righe troppo lunghe
 98          if(*buffer == '!') {
 99              struct pila *trovato = search_prefix(history,
                 buffer+1);
100              if(trovato) {
101                  printf("input: '%s': eseguo: %s\n", buffer,
                     trovato->cmd);
102                  if(trovato != history) {
```

```
103                     delete_node(history, trovato);
104                     trovato->next = history;
105                     history = trovato;
106                 }
107             } else {
108                 printf("input: '%s': comando non trovato\n",
                    buffer);
109             }
110         } else {
111             struct pila *trovato = search_cmd(history, buffer);
112             if(trovato) {
113                 /*
114                  * Se un comando è già stato dato
115                  * non duplicarlo ma spostarlo in prima posizione
116                  */
117                 if(trovato != history) {
118                     delete_node(history, trovato);
119                     trovato->next = history;
120                     history = trovato;
121                 }
122             } else {
123                 /* altrimenti lo aggiungo in testa */
124                 history = push(history, buffer);
125             }
126         }
127     }
```

```
128         if(fout) {
129             /*
130              * deve essere l'ultima operazione del programma
131              * viene modificato il valore di history
132              */
133             while(history) {
134                 fprintf(fout, "%s\n", history->cmd);
135                 history = history->next;
136             }
137             /* a questo punto history è sempre NULL */
138         }
139         return 0;
140     }
141
```