計算機程式語言

教課教授 : 謝東儒

助教 :蔡詠聿、吳品頤

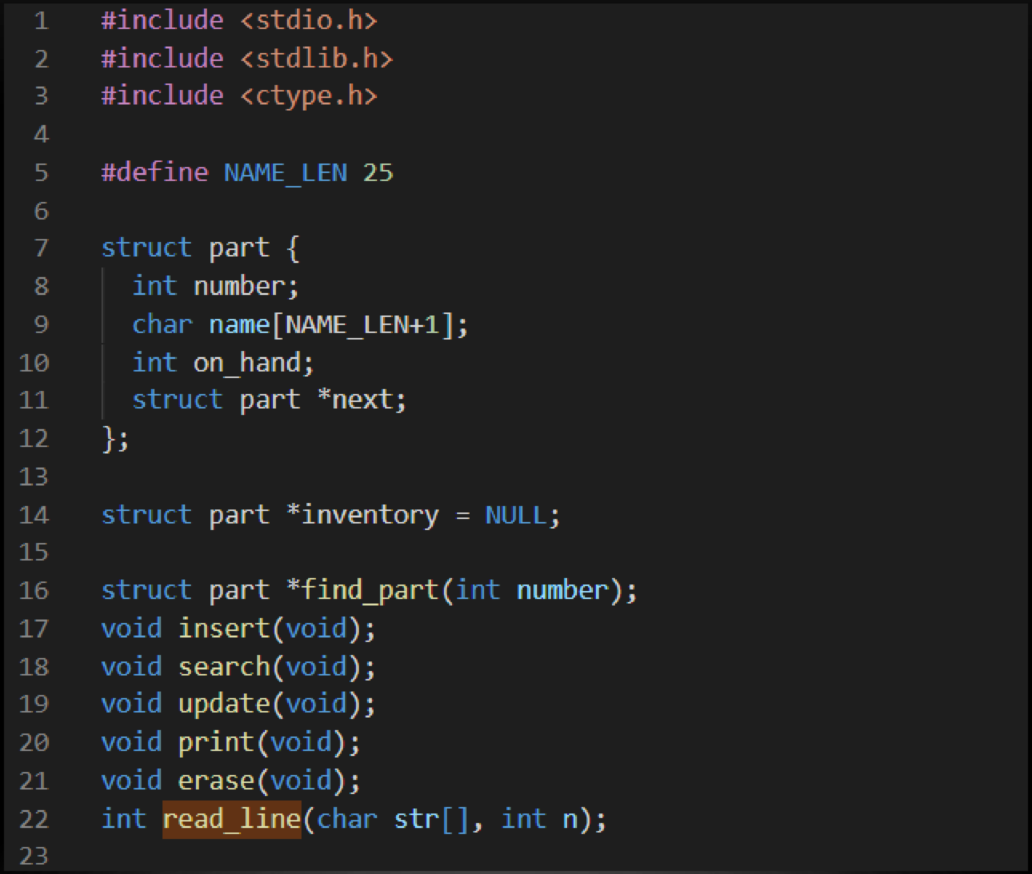
Chapter 17_project 3

Modify the inventory2.c program of Section 17.5 by adding an e(erase) command that allows the user to remove a part from the database.

請確實實作link list

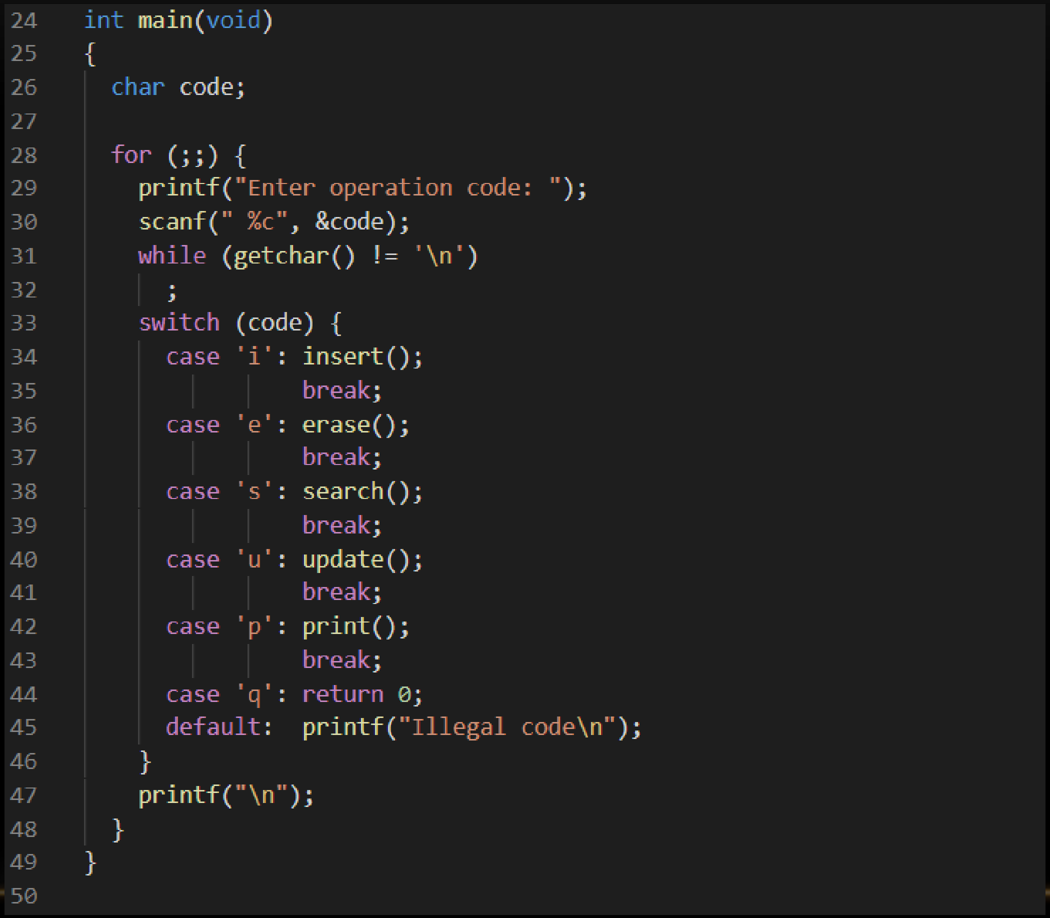
本次作業只會有一個.c檔，不會有其他的.h或.c檔

Solution



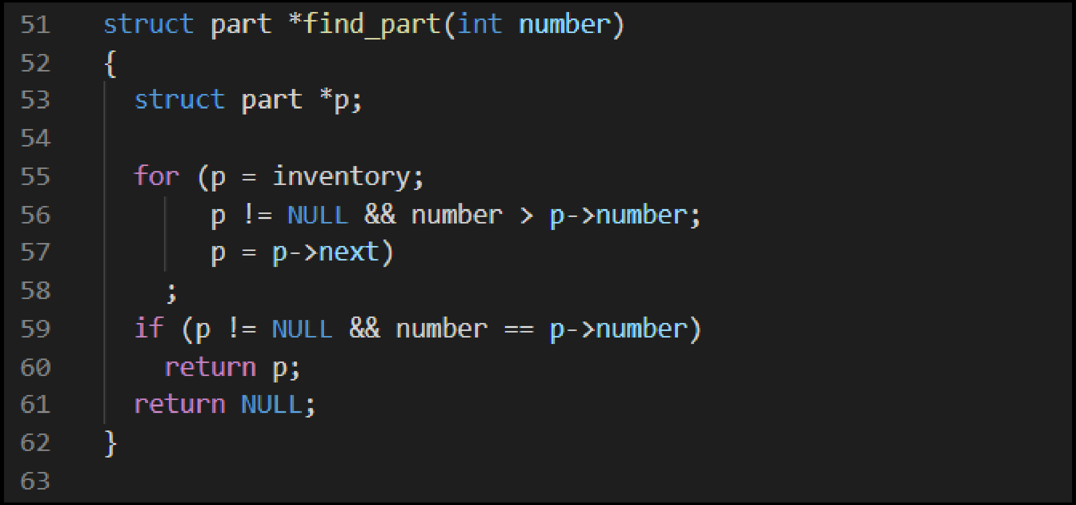
```c
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>

#define NAME_LEN 25

struct part {
  int number;
  char name[NAME_LEN+1];
  int on_hand;
  struct part *next;
};

struct part *inventory = NULL;

struct part *find_part(int number);
void insert(void);
void search(void);
void update(void);
void print(void);
void erase(void);
int read_line(char str[], int n);

```

Solution



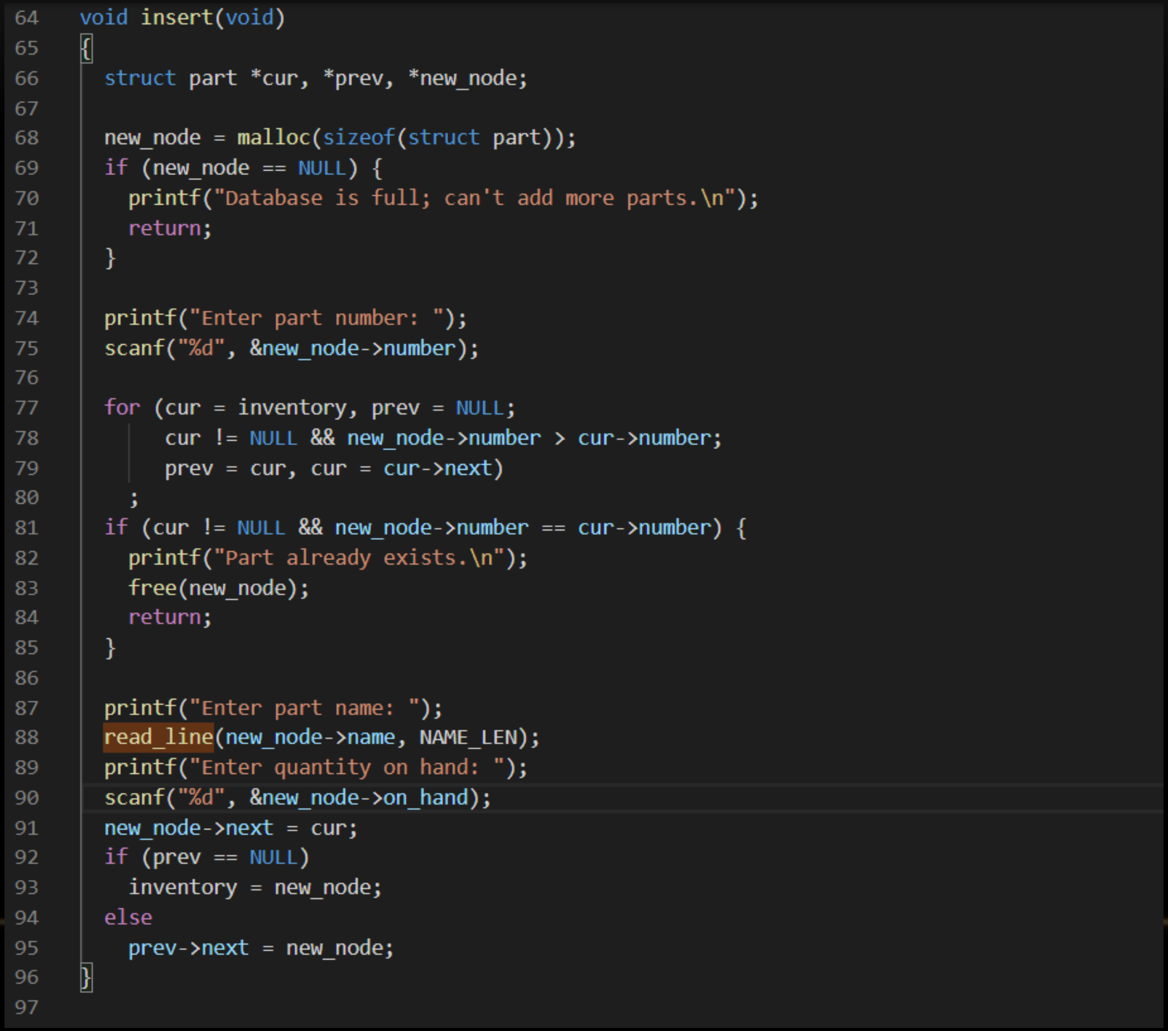
```c
int main(void)
{
  char code;

  for (;;) {
    printf("Enter operation code: ");
    scanf(" %c", &code);
    while (getchar() != '\n')
      ;
    switch (code) {
      case 'i': insert();
                break;
      case 'e': erase();
                break;
      case 's': search();
                break;
      case 'u': update();
                break;
      case 'p': print();
                break;
      case 'q': return 0;
      default:  printf("Illegal code\n");
    }
    printf("\n");
  }
}
```

Solution



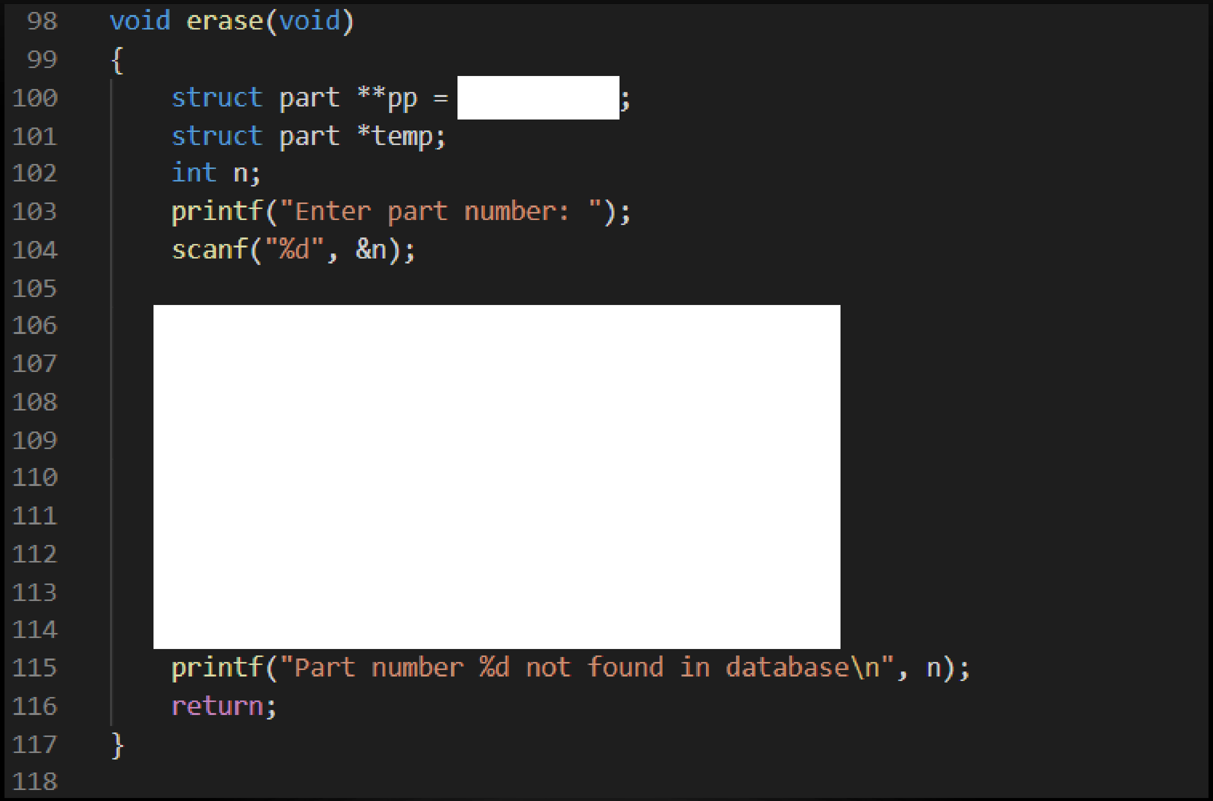
```c
struct part *find_part(int number)
{
  struct part *p;

  for (p = inventory;
       p != NULL && number > p->number;
       p = p->next)
    ;
  if (p != NULL && number == p->number)
    return p;
  return NULL;
}

```

Solution



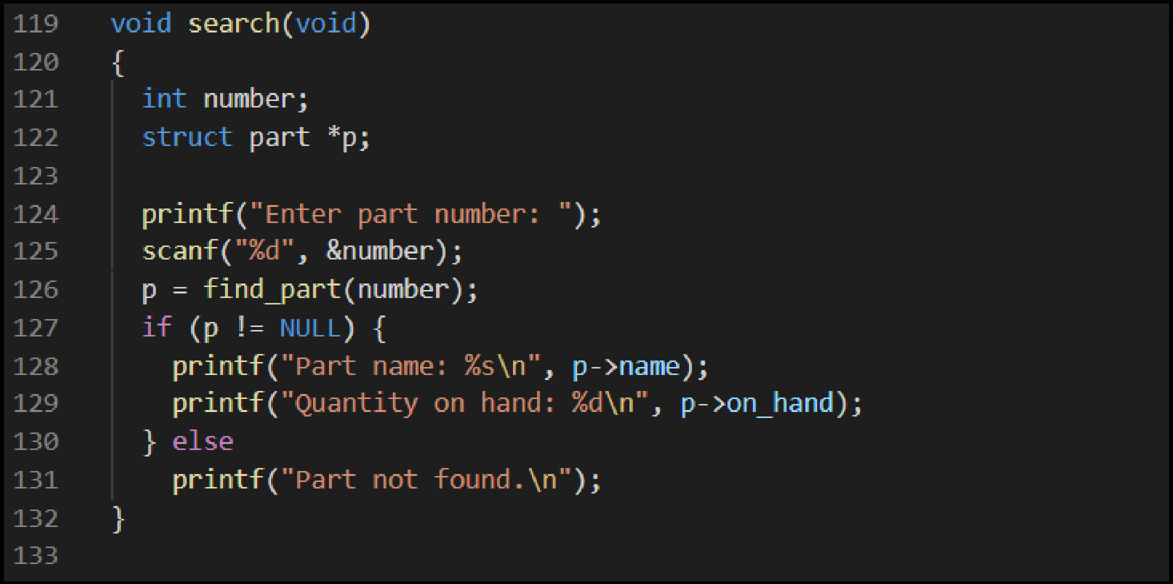
```c
void insert(void)
{
  struct part *cur, *prev, *new_node;

  new_node = malloc(sizeof(struct part));
  if (new_node == NULL) {
    printf("Database is full; can't add more parts.\n");
    return;
  }

  printf("Enter part number: ");
  scanf("%d", &new_node->number);

  for (cur = inventory, prev = NULL;
       cur != NULL && new_node->number > cur->number;
       prev = cur, cur = cur->next)
    ;
  if (cur != NULL && new_node->number == cur->number) {
    printf("Part already exists.\n");
    free(new_node);
    return;
  }

  printf("Enter part name: ");
  read_line(new_node->name, NAME_LEN);
  printf("Enter quantity on hand: ");
  scanf("%d", &new_node->on_hand);
  new_node->next = cur;
  if (prev == NULL)
    inventory = new_node;
  else
    prev->next = new_node;
}
```

Solution



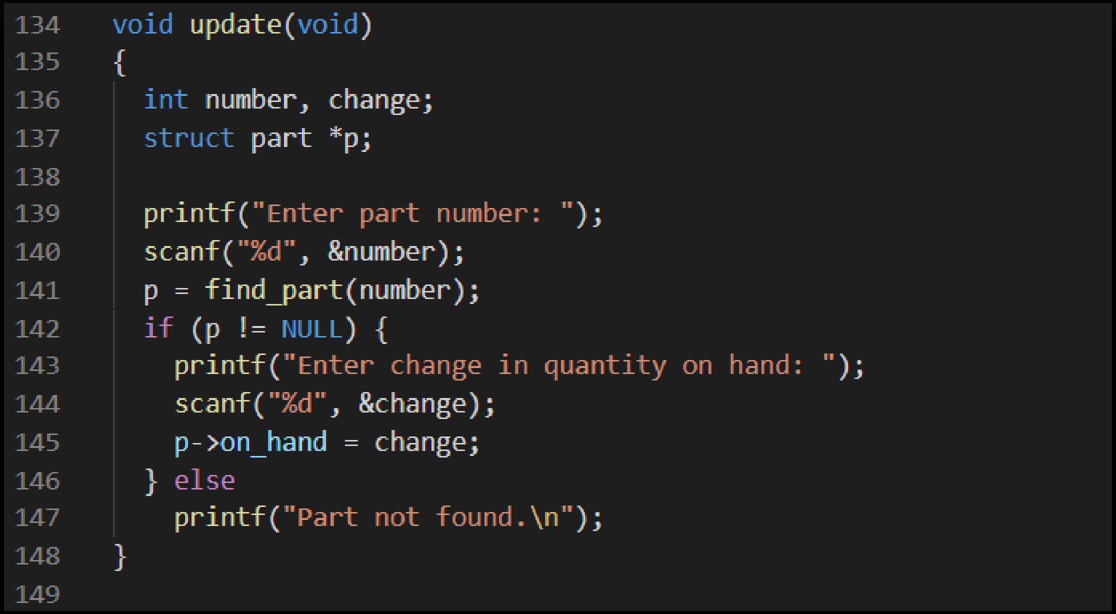
```
void erase(void)
{
    struct part **pp =            ;
    struct part *temp;
    int n;
    printf("Enter part number: ");
    scanf("%d", &n);










    printf("Part number %d not found in database\n", n);
    return;
}

```

Solution



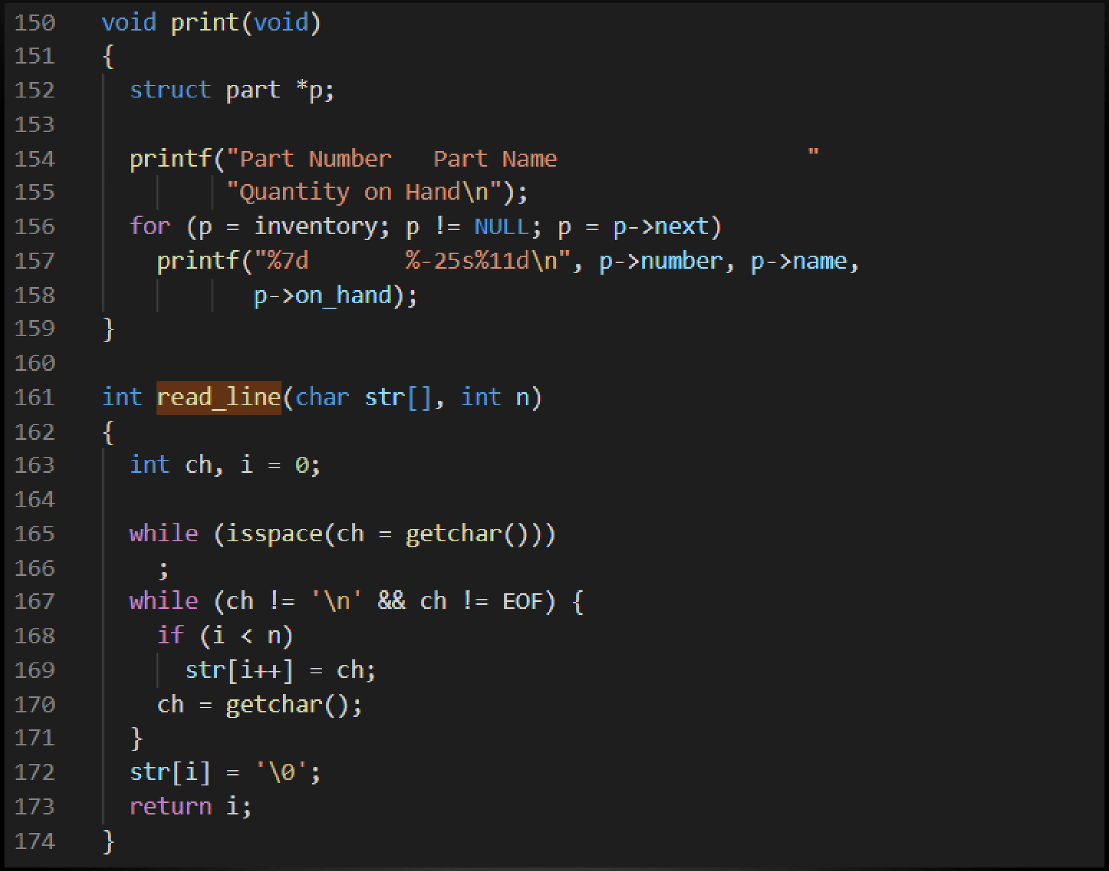
```c
void search(void)
{
  int number;
  struct part *p;

  printf("Enter part number: ");
  scanf("%d", &number);
  p = find_part(number);
  if (p != NULL) {
    printf("Part name: %s\n", p->name);
    printf("Quantity on hand: %d\n", p->on_hand);
  } else
    printf("Part not found.\n");
}
```

Solution



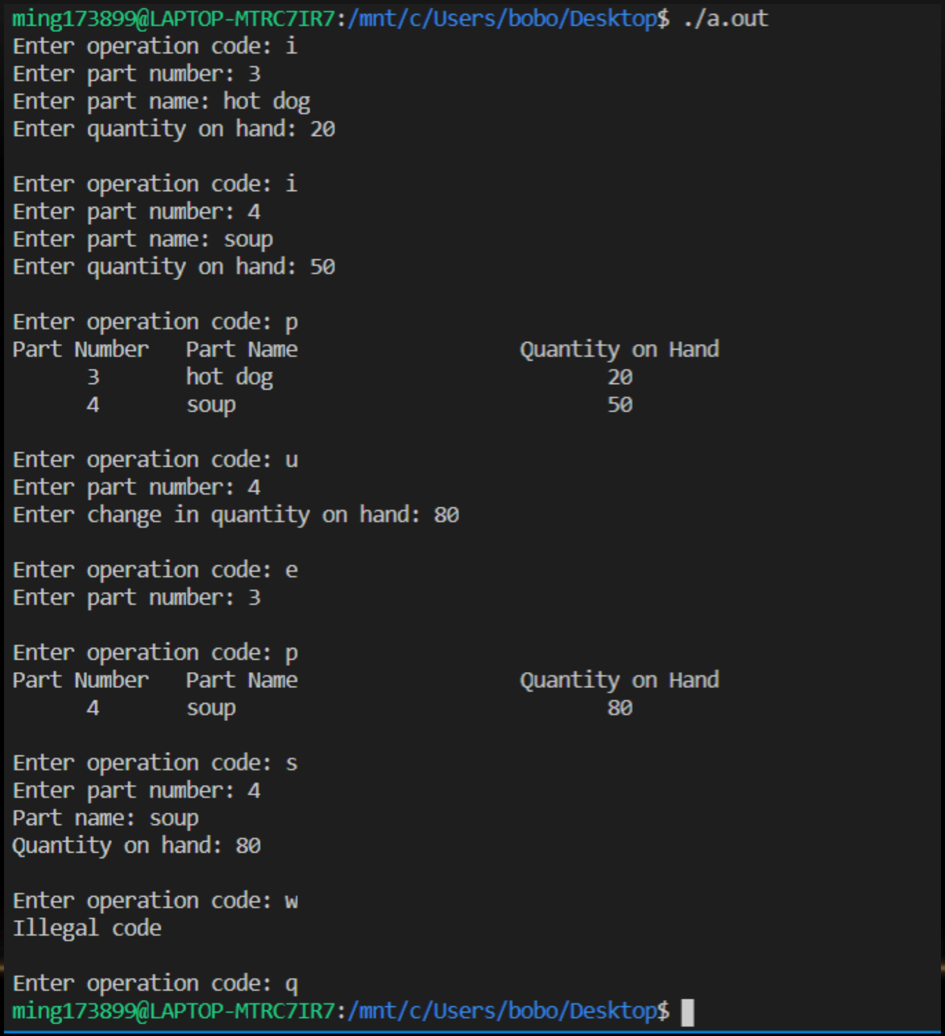
```
void update(void)
{
  int number, change;
  struct part *p;

  printf("Enter part number: ");
  scanf("%d", &number);
  p = find_part(number);
  if (p != NULL) {
    printf("Enter change in quantity on hand: ");
    scanf("%d", &change);
    p->on_hand = change;
  } else
    printf("Part not found.\n");
}
```

Solution



```c
void print(void)
{
  struct part *p;

  printf("Part Number   Part Name                  "
         "Quantity on Hand\n");
  for (p = inventory; p != NULL; p = p->next)
    printf("%7d       %-25s%11d\n", p->number, p->name,
           p->on_hand);
}

int read_line(char str[], int n)
{
  int ch, i = 0;

  while (isspace(ch = getchar()))
    ;
  while (ch != '\n' && ch != EOF) {
    if (i < n)
      str[i++] = ch;
    ch = getchar();
  }
  str[i] = '\0';
  return i;
}
```

Example



```
ming173899@LAPTOP-MTRC7IR7:/mnt/c/Users/bobo/Desktop$ ./a.out
Enter operation code: i
Enter part number: 3
Enter part name: hot dog
Enter quantity on hand: 20

Enter operation code: i
Enter part number: 4
Enter part name: soup
Enter quantity on hand: 50

Enter operation code: p
Part Number   Part Name                  Quantity on Hand
      3       hot dog                           20
      4       soup                              50

Enter operation code: u
Enter part number: 4
Enter change in quantity on hand: 80

Enter operation code: e
Enter part number: 3

Enter operation code: p
Part Number   Part Name                  Quantity on Hand
      4       soup                              80

Enter operation code: s
Enter part number: 4
Part name: soup
Quantity on hand: 80

Enter operation code: w
Illegal code

Enter operation code: q
ming173899@LAPTOP-MTRC7IR7:/mnt/c/Users/bobo/Desktop$
```

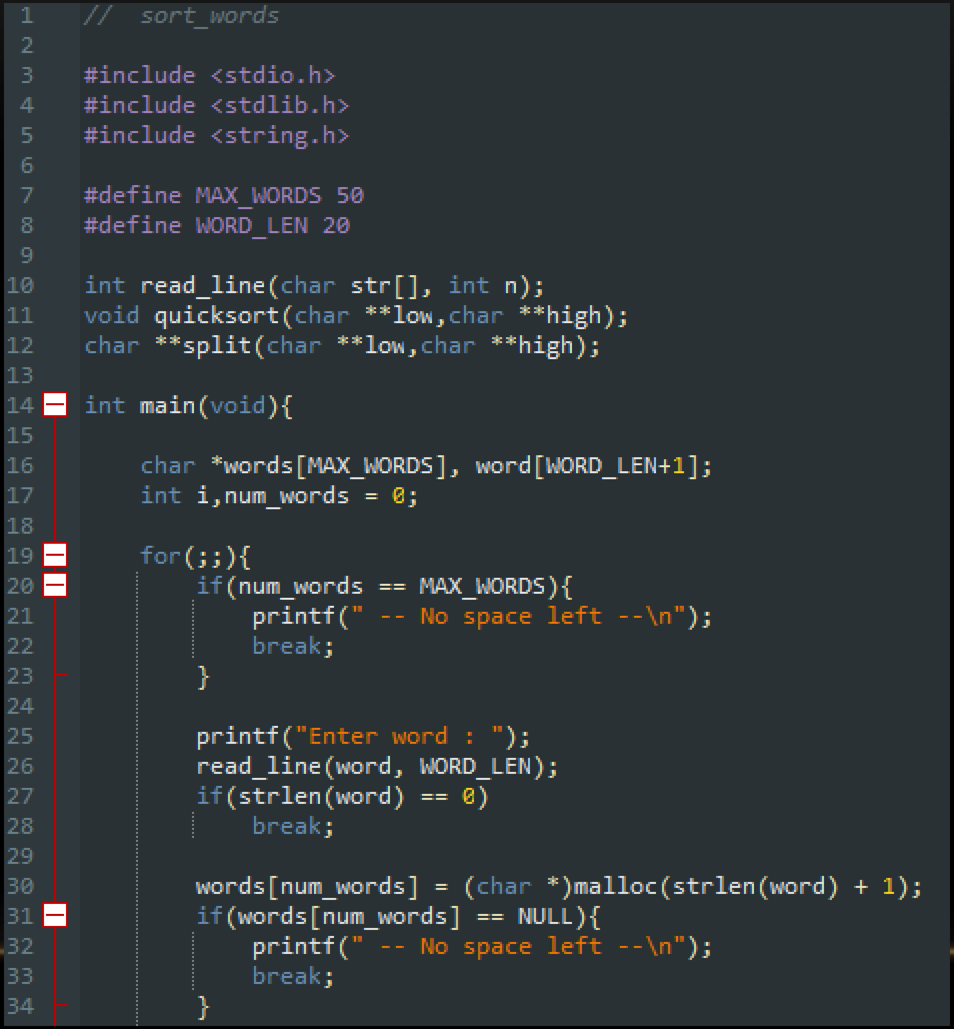
Chapter 17_project 5

Write a program that sorts a series of words entered by the user:

Assume that each word is no more than 20 characters long. Stop reading when the user enters an empty word (i.e., presses Enter without entering a word). Store each word in a dynamically allocated string, using an array of pointers to keep track of the strings, as in the remind2.c program (Section 17.2). After all words have been read, sort the array (using any sorting technique) and then use a loop to print the words in sorted order. Hint: Use the read_line function to read each word. as in remind2 . c.

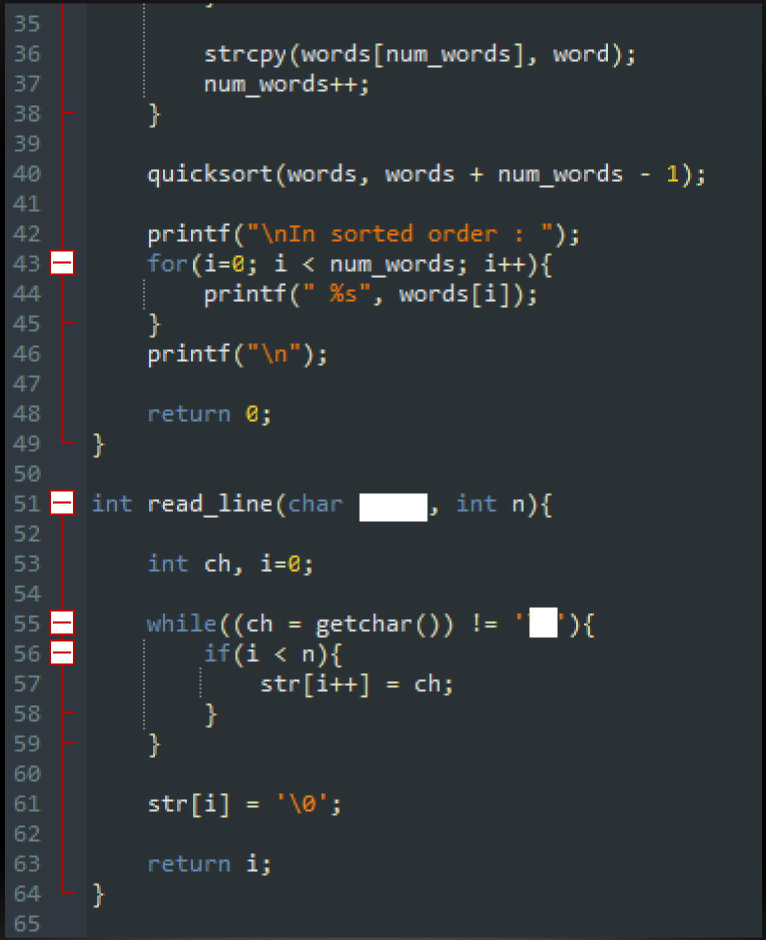
請不要使用qsort()

Solution



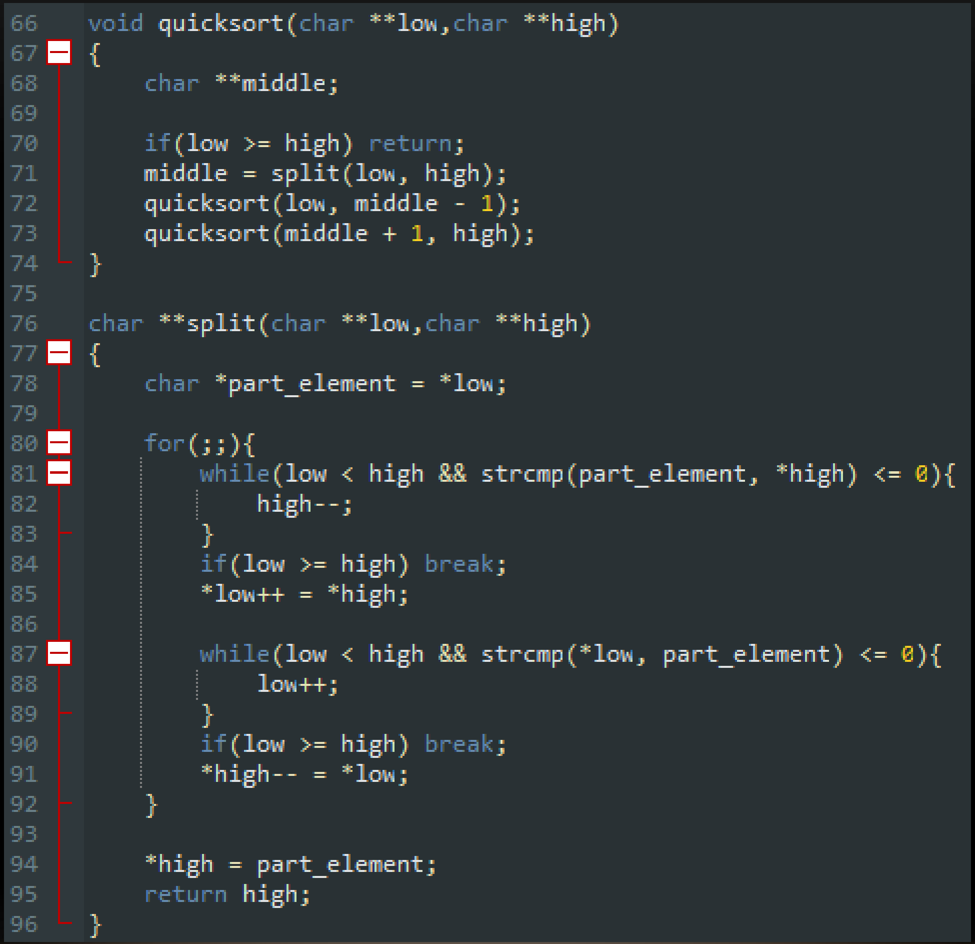
```c
//  sort_words

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

#define MAX_WORDS 50
#define WORD_LEN 20

int read_line(char str[], int n);
void quicksort(char **low,char **high);
char **split(char **low,char **high);

int main(void){

    char *words[MAX_WORDS], word[WORD_LEN+1];
    int i,num_words = 0;

    for(;;){
        if(num_words == MAX_WORDS){
            printf(" -- No space left --\n");
            break;
        }

        printf("Enter word : ");
        read_line(word, WORD_LEN);
        if(strlen(word) == 0)
            break;

        words[num_words] = (char *)malloc(strlen(word) + 1);
        if(words[num_words] == NULL){
            printf(" -- No space left --\n");
            break;
        }
```

Solution



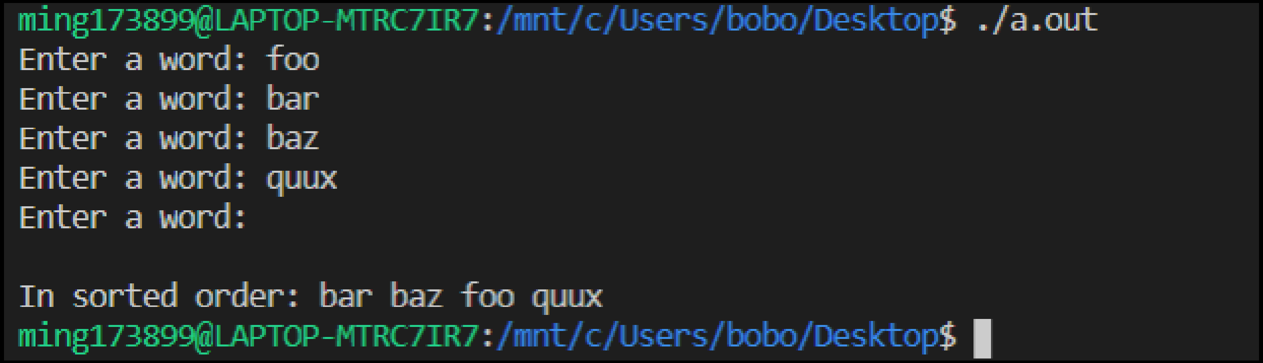
```
        }

        strcpy(words[num_words], word);
        num_words++;
    }

    quicksort(words, words + num_words - 1);

    printf("\nIn sorted order : ");
    for(i=0; i < num_words; i++){
        printf(" %s", words[i]);
    }
    printf("\n");

    return 0;
}

int read_line(char      , int n){

    int ch, i=0;

    while((ch = getchar()) != '  '){
        if(i < n){
            str[i++] = ch;
        }
    }

    str[i] = '\0';

    return i;
}
```

Solution



```c
void quicksort(char **low,char **high)
{
    char **middle;

    if(low >= high) return;
    middle = split(low, high);
    quicksort(low, middle - 1);
    quicksort(middle + 1, high);
}

char **split(char **low,char **high)
{
    char *part_element = *low;

    for(;;){
        while(low < high && strcmp(part_element, *high) <= 0){
            high--;
        }
        if(low >= high) break;
        *low++ = *high;

        while(low < high && strcmp(*low, part_element) <= 0){
            low++;
        }
        if(low >= high) break;
        *high-- = *low;
    }

    *high = part_element;
    return high;
}
```

Example

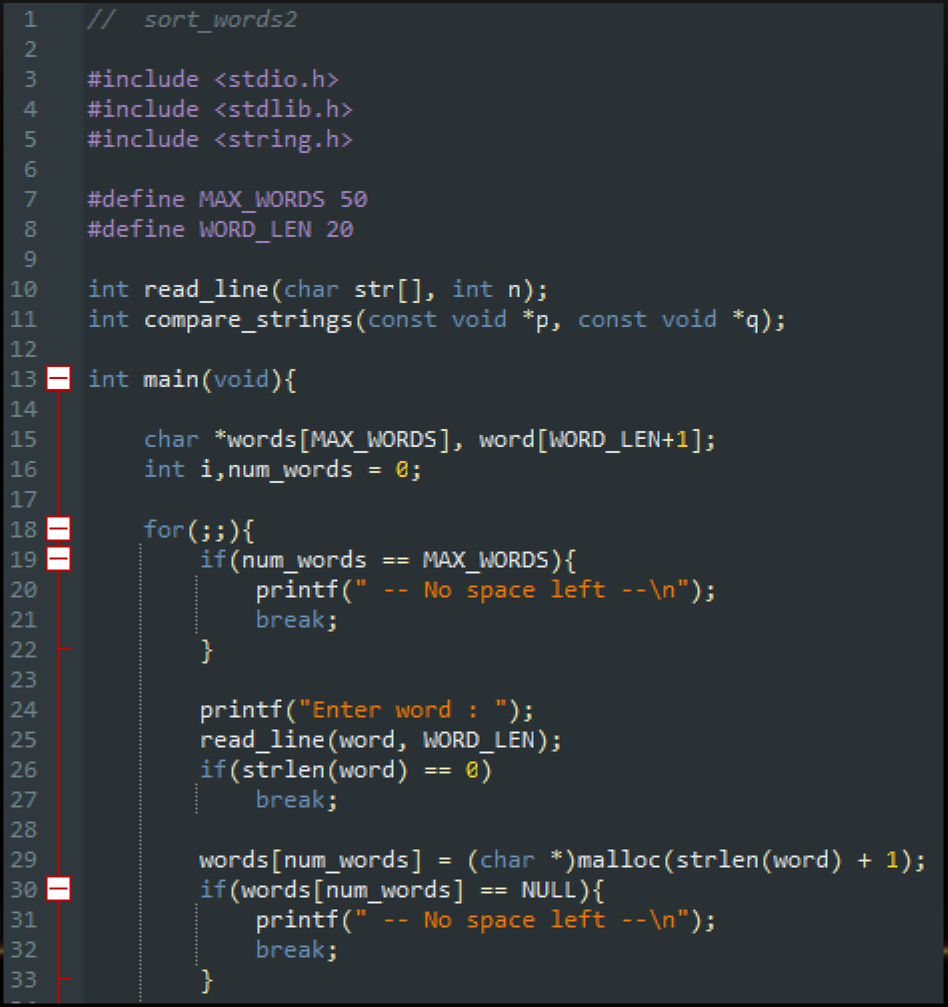


```
ming173899@LAPTOP-MTRC7IR7:/mnt/c/Users/bobo/Desktop$ ./a.out
Enter a word: foo
Enter a word: bar
Enter a word: baz
Enter a word: quux
Enter a word:

In sorted order: bar baz foo quux
ming173899@LAPTOP-MTRC7IR7:/mnt/c/Users/bobo/Desktop$
```

Chapter 17_project 6

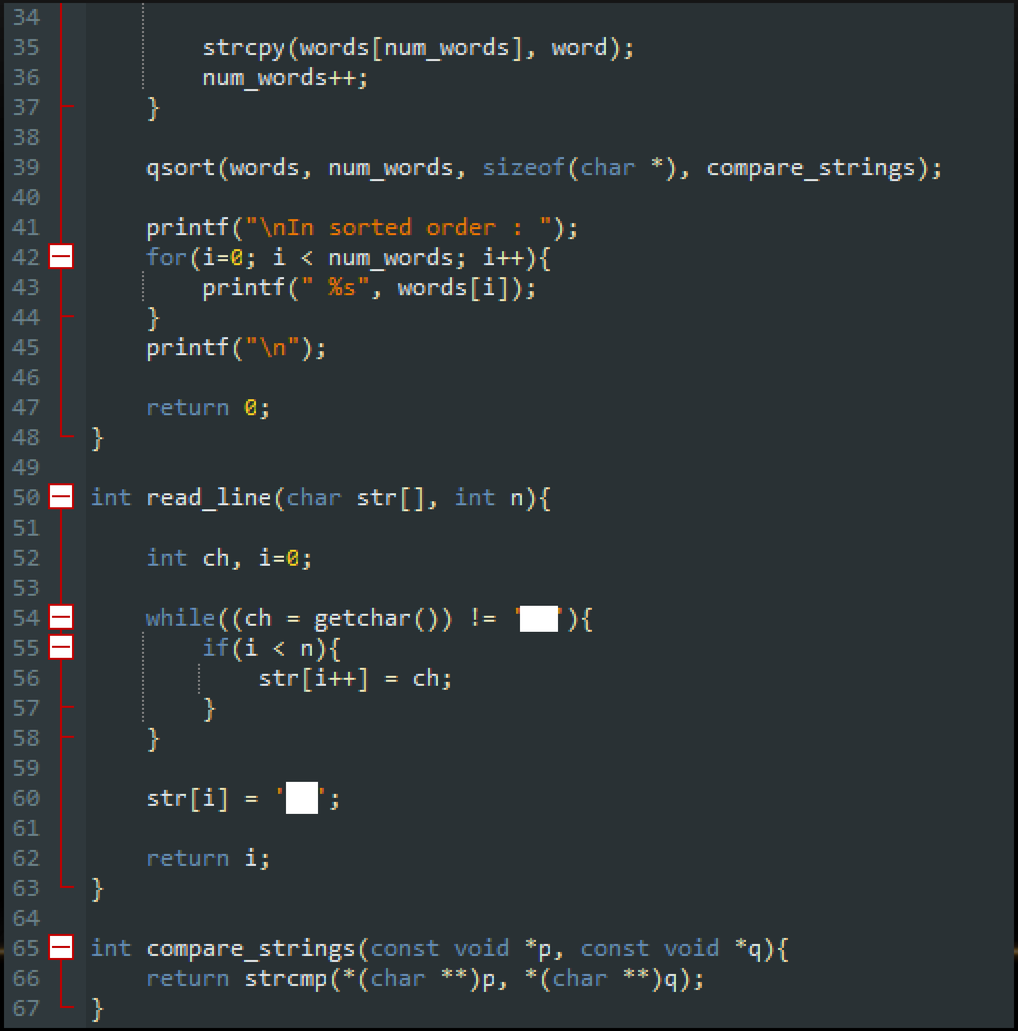
Modify Programming Project 5 so that it uses qsort to sort the array of pointers.

Solution



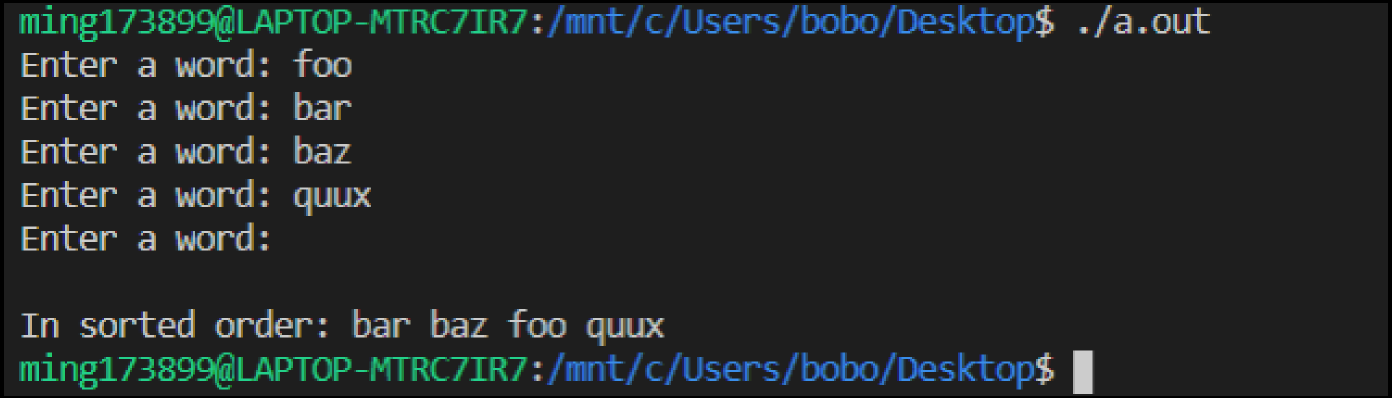
```c
//  sort_words2

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

#define MAX_WORDS 50
#define WORD_LEN 20

int read_line(char str[], int n);
int compare_strings(const void *p, const void *q);

int main(void){

    char *words[MAX_WORDS], word[WORD_LEN+1];
    int i,num_words = 0;

    for(;;){
        if(num_words == MAX_WORDS){
            printf(" -- No space left --\n");
            break;
        }

        printf("Enter word : ");
        read_line(word, WORD_LEN);
        if(strlen(word) == 0)
            break;

        words[num_words] = (char *)malloc(strlen(word) + 1);
        if(words[num_words] == NULL){
            printf(" -- No space left --\n");
            break;
        }
```

Solution



```c

        strcpy(words[num_words], word);
        num_words++;
    }

    qsort(words, num_words, sizeof(char *), compare_strings);

    printf("\nIn sorted order : ");
    for(i=0; i < num_words; i++){
        printf(" %s", words[i]);
    }
    printf("\n");

    return 0;
}

int read_line(char str[], int n){

    int ch, i=0;

    while((ch = getchar()) != '  '){
        if(i < n){
            str[i++] = ch;
        }
    }

    str[i] = '  ';

    return i;
}

int compare_strings(const void *p, const void *q){
    return strcmp(*(char **)p, *(char **)q);
}
```

Example

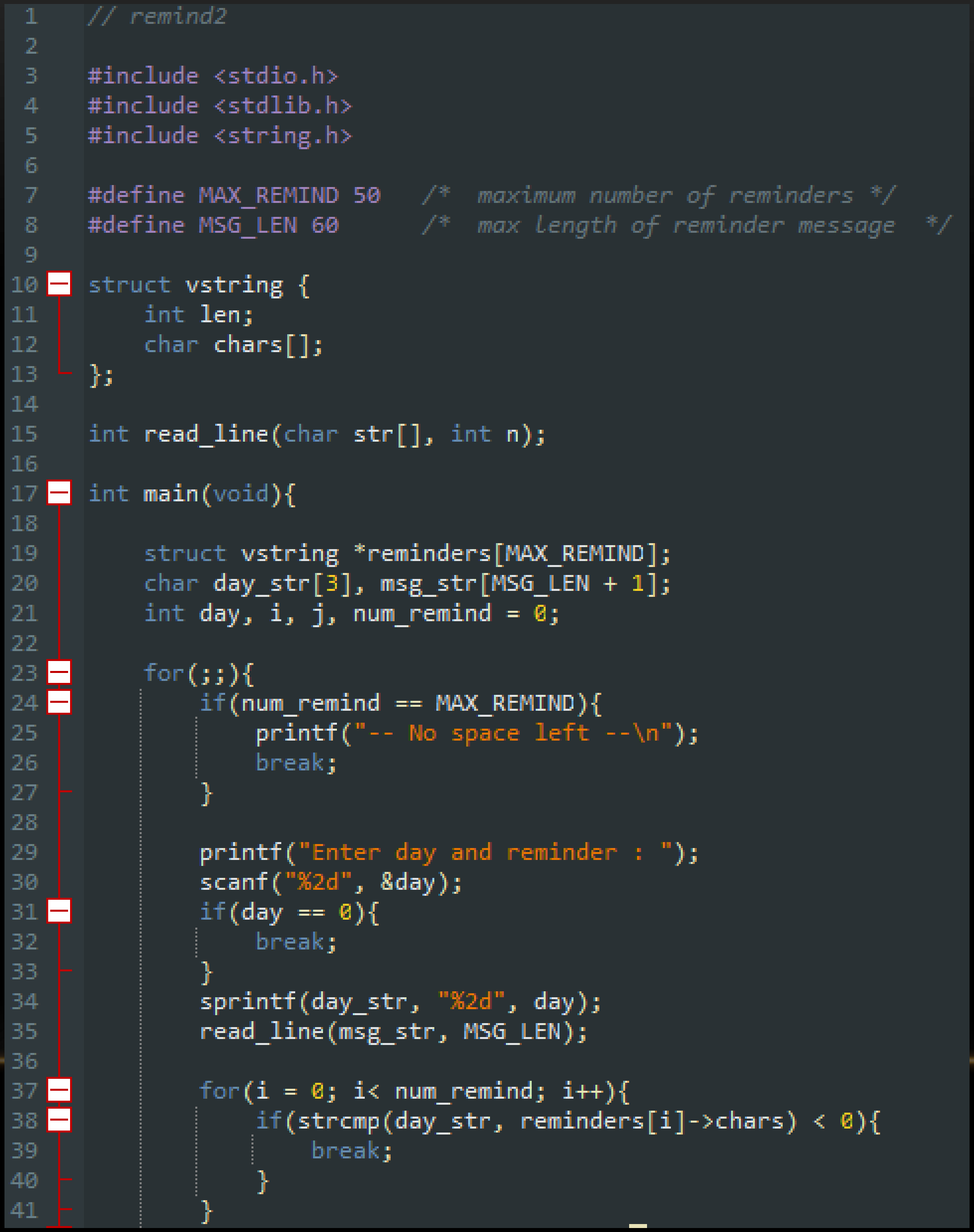


```
ming173899@LAPTOP-MTRC7IR7:/mnt/c/Users/bobo/Desktop$ ./a.out
Enter a word: foo
Enter a word: bar
Enter a word: baz
Enter a word: quux
Enter a word:

In sorted order: bar baz foo quux
ming173899@LAPTOP-MTRC7IR7:/mnt/c/Users/bobo/Desktop$
```

Chapter 17_project 7

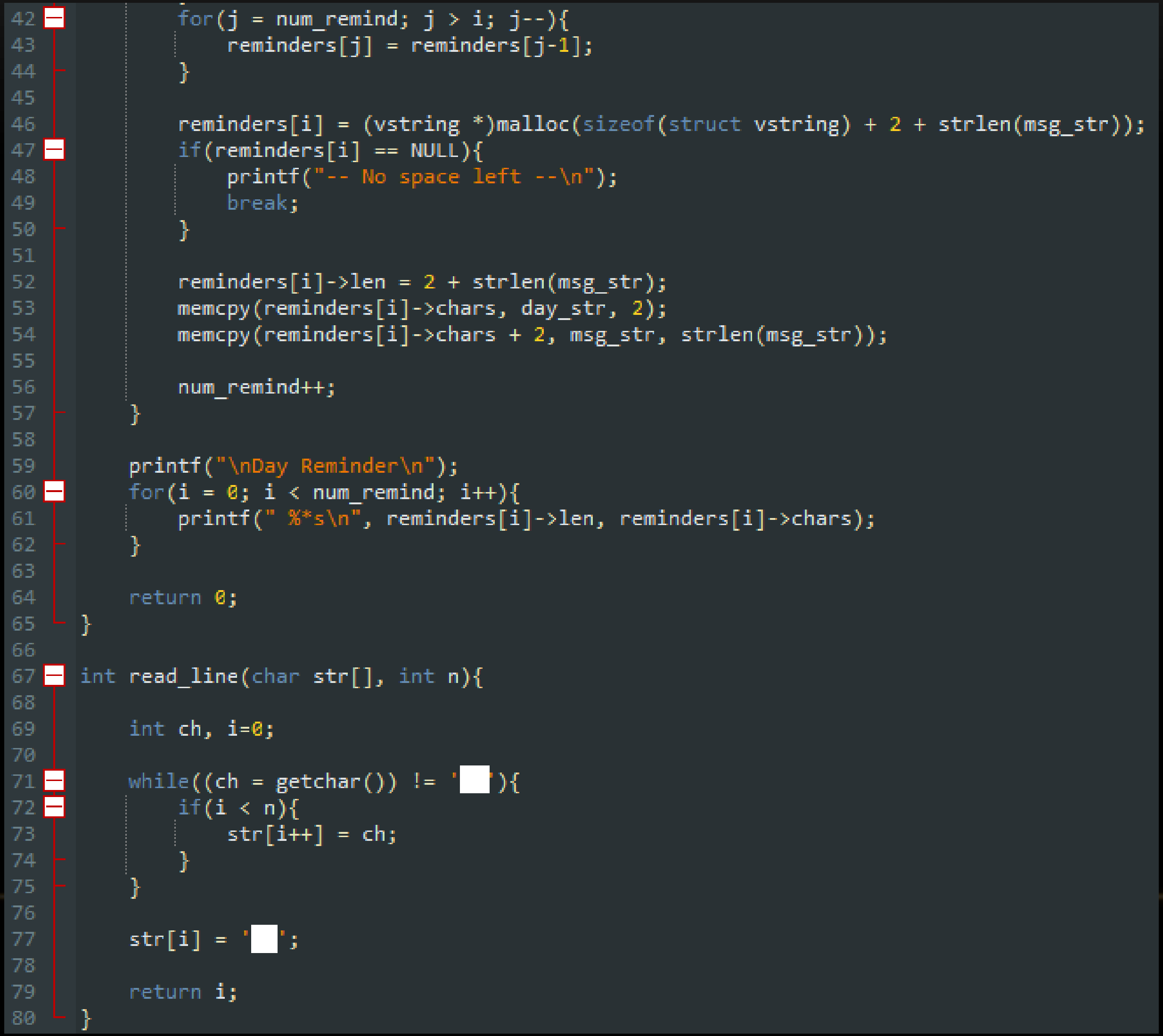
(C99) Modify the remind2.c program of Section 17.2 so that each element of the reminders array is a pointer to a vstring structure (see Section 17.9) rather than a pointer to an ordinary string.

Solution



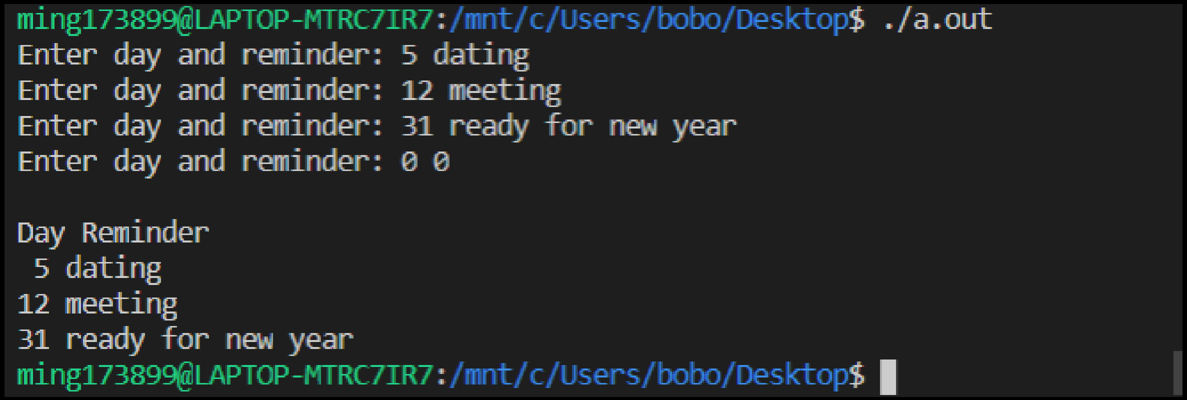
```c
// remind2

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

#define MAX_REMIND 50   /*  maximum number of reminders */
#define MSG_LEN 60      /*  max length of reminder message  */

struct vstring {
    int len;
    char chars[];
};

int read_line(char str[], int n);

int main(void){

    struct vstring *reminders[MAX_REMIND];
    char day_str[3], msg_str[MSG_LEN + 1];
    int day, i, j, num_remind = 0;

    for(;;){
        if(num_remind == MAX_REMIND){
            printf("-- No space left --\n");
            break;
        }

        printf("Enter day and reminder : ");
        scanf("%2d", &day);
        if(day == 0){
            break;
        }
        sprintf(day_str, "%2d", day);
        read_line(msg_str, MSG_LEN);

        for(i = 0; i< num_remind; i++){
            if(strcmp(day_str, reminders[i]->chars) < 0){
                break;
            }
        }
```

Solution



```
        for(j = num_remind; j > i; j--){
            reminders[j] = reminders[j-1];
        }

        reminders[i] = (vstring *)malloc(sizeof(struct vstring) + 2 + strlen(msg_str));
        if(reminders[i] == NULL){
            printf("-- No space left --\n");
            break;
        }

        reminders[i]->len = 2 + strlen(msg_str);
        memcpy(reminders[i]->chars, day_str, 2);
        memcpy(reminders[i]->chars + 2, msg_str, strlen(msg_str));

        num_remind++;
    }

    printf("\nDay Reminder\n");
    for(i = 0; i < num_remind; i++){
        printf(" %*s\n", reminders[i]->len, reminders[i]->chars);
    }

    return 0;
}

int read_line(char str[], int n){

    int ch, i=0;

    while((ch = getchar()) != '□'){
        if(i < n){
            str[i++] = ch;
        }
    }

    str[i] = '□';

    return i;
}
```

Example



```
ming173899@LAPTOP-MTRC7IR7:/mnt/c/Users/bobo/Desktop$ ./a.out
Enter day and reminder: 5 dating
Enter day and reminder: 12 meeting
Enter day and reminder: 31 ready for new year
Enter day and reminder: 0 0

Day Reminder
 5 dating
12 meeting
31 ready for new year
ming173899@LAPTOP-MTRC7IR7:/mnt/c/Users/bobo/Desktop$
```