1. （線性搜尋）修改圖 6.18 的程式，改使用遞迴函式 linearSearch 來對陣列執行線性搜尋。此函式會接收一個整數陣列、此陣列的大小以及搜尋關鍵字為其引數。若找到搜尋關鍵字的話，傳回陣列的索引，否則傳回-1。

```c
// Fig. 6.18: fig06_18.c
// Linear search of an array.
#include <stdio.h>
#define SIZE 100

// function prototype
size_t linearSearch(const int array[], int key, size_t size);

// function main begins program execution
int main(void)
{
   int a[SIZE]; // create array a

   // create some data
   for (size_t x = 0; x < SIZE; ++x) {
      a[x] = 2 * x;
   }

   printf("Enter integer search key: ");
   int searchKey; // value to locate in array a
   scanf("%d", &searchKey);


   // attempt to locate searchKey in array a
   size_t index = linearSearch(a, searchKey, SIZE);

   // display results
   if (index != -1) {
      printf("Found value at index %d\n", index);
   }
   else {
      puts("Value not found");
   }
}

```

```
size_t linearSearch(const int array[], int key, size_t size)
{
   // loop through array
   for (size_t n = 0; n < size; ++n) {

      if (array[n] == key) {
         return n; // return location of key
      }
   }

   return -1; // key not found
}
```

```
Enter integer search key: 36
Found value at index 18
```

```
Enter integer search key: 37
Value not found
```

2. （選擇排序）一維陣列的選擇排序演算法有以下步驟：

   a) 找到陣列中的最小值。

   b) 將它與陣列第一個位置的值交換。

   c) 對於陣列的其餘部分重複上述步驟，從第二個位置開始並每次前進。

最後整個陣列分為兩部分：已經排序之項目的子陣列，其由左向右構建，而且在開始時找到；另一個是仍需要被排序之項目的子陣列，佔用陣列的剩餘部分。請編寫一個程式，使用這種演算法對 10 個整數的陣列進行排序。

1. ***Linear Search)*** Modify the program of Fig. 6.18 to use a recursive linearSearch function to perform the linear search of the array. The function should receive an integer array, the size of the array and the search key as arguments. If the search key is found, return the array index; otherwise, return –1.

```c
// Fig. 6.18: fig06_18.c
// Linear search of an array.
#include <stdio.h>
#define SIZE 100

// function prototype
size_t linearSearch(const int array[], int key, size_t size);

// function main begins program execution
int main(void)
{
   int a[SIZE]; // create array a

   // create some data
   for (size_t x = 0; x < SIZE; ++x) {
      a[x] = 2 * x;
   }

   printf("Enter integer search key: ");
   int searchKey; // value to locate in array a
   scanf("%d", &searchKey);


   // attempt to locate searchKey in array a
   size_t index = linearSearch(a, searchKey, SIZE);

   // display results
   if (index != -1) {
      printf("Found value at index %d\n", index);
   }
   else {
      puts("Value not found");
   }
}

```

```cpp
size_t linearSearch(const int array[], int key, size_t size)
{
   // loop through array
   for (size_t n = 0; n < size; ++n) {

      if (array[n] == key) {
         return n; // return location of key
      }
   }

   return -1; // key not found
}
```

```
Enter integer search key: 36
Found value at index 18
```

```
Enter integer search key: 37
Value not found
```

2. ***(Selection Sort)*** A *selection sort algorithm* for a one-dimensional array has the following steps:
   a) The smallest value in the array is found.
   b) It is swapped with the value in the first position of the array.
   c) The above steps are repeated for the rest of the array starting at the second position and advancing each time.

   Eventually the entire array is divided into two parts: the sub-array of items already sorted which is built up from left to right and is found at the beginning, and the sub-array of items remaining to be sorted, occupying the remainder of the array. Write a program that sorts an array of 10 integers using this algorithm.