Bài tập thực hành
Cấu trúc dữ liệu và thuật toán
Tìm kiếm tuyến tính, tìm kiếm nhị phân (A)

1. Bài 1

Hoàn thành chương trình thực hiện giải thuật tìm kiếm tuyến tính và tìm kiếm nhị phân.

a) Hoàn thành hàm print_array(int K[], int i1, int i2) thực hiện đưa các phần tử từ chỉ số i1 tới phần tử chỉ số i2 trong mảng K ra màn hình

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

void print_array(int K[], int i1, int i2) {
  int i;
  for (i = ___ ; i <= ___ ; i++) {
        printf("%d ", K[i]);
  }
}
```

b) Hoàn thành hàm int linearSearch(int list[], int size, int target) dùng giải thuật tìm kiếm tuyến tính để tìm phần tử target trong mảng list có size phần tử. Hàm này trả về chỉ số vị trí của phần tử target nếu có phần tử target trong mảng, nếu không thì trả về -1. Giả sử các phần tử trong mảng có giá trị khác nhau.

```
int linearSearch(int list[], int size, int target) {
  int i;
  for (i = 0; i < size; i++)
        if (list[i] == target){
              return ___ ;
        }
```

```
  return ___;
}
```

c) Hoàn thành hàm int binarySearch(int list[], int size, int target) dùng giải thuật tìm kiếm nhị phân để tìm phần tử target trong mảng list có size phần tử. Hàm này trả về chỉ số vị trí của phần tử target nếu có phần tử target trong mảng, nếu không thì trả về -1. Giả sử các phần tử trong mảng có giá trị khác nhau.

```
int binarySearch(int list[], int size, int target) {
  int lower = 0, upper = ___ , mid;
  while (lower <= upper) {
        mid = (upper + lower)/2;
        if (target < list[mid])
              upper = ___ ;
        else if (target > list[mid])
              lower = ___ ;
        else return ___ ;
  }
  return ___ ;
}
```

d) Hoàn thành hàm main() để kiểm tra giải thuật đã viết.

```
int main(void) {
  int k[7] = {40, 66, 22, 55, 11, 88};
  int b[7] = {11, 22, 40, 55, 66, 88};
  int n = 6; // so phan tu cua mang

  printf("\nMang da cho: ");
  print_array(k, 0, ___ );

  int target = 22;
  int r = linearSearch(k, 6 , 22);
  if (r == ___ )
        printf("\nKhong ton tai phan tu %d trong mang", target);
```

```
  else printf("\nChi so cua vi tri phan tu %d trong mang la: %d", ___ ,  ___ );

  printf("\nMang da cho: ");
  print_array(b, 0, n-1);
  target = 22;
  r = binarySearch(b, 6, 22);
  if (r == -1)
        printf("\nKhong ton tai phan tu %d trong mang", target);
  else printf("\nChi so cua vi tri phan tu %d trong mang la: %d", ___, ___ );

  return 0;
}
```

2. Bài 2

Cho mảng K đã được **sắp xếp không giảm**. Viết thuật toán tìm kiếm nhị phân hàm đếm số phần tử bằng X trong mảng K[0..n] với độ phức tạp O(logn).

Ví dụ:

Input: K[] = {1, 1, 2, 2, 2, 2, 3}, X = 2

Output: 4

a) Hoàn thành hàm print_array(int K[], int i1, int i2) thực hiện đưa các phần tử từ chỉ số i1 tới phần tử chỉ số i2 trong mảng K ra màn hình

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

void print_array(int K[], int i1, int i2) {
  int i;
  for (i = ___ ; i <= ___ ; i++) {
        printf("%d ", K[i]);
  }
}
```

b) Hoàn thành hàm trả về chỉ số của phần tử x xuất hiện đầu tiên trong mảng, -1 nếu không tồn tại. n là số phần tử của mảng arr, low là chỉ số phần tử đầu tiên của mảng, high là chỉ số phần tử cuối cùng của mảng.

```
int first(int arr[], int low, int high, int x, int n)
{
 if(high >= low)
 {
  int mid = (low + high)/2;
  if( ( mid == 0 || x > ___ ) && arr[mid] == x)
    return mid;
  else if(x > arr[mid])
    return first(arr, ___ , high, x, n);
  else
    return first(arr, low, ___ , x, n);
 }
 return -1;
}
```

c) Hoàn thành hàm trả về chỉ số của phần tử x xuất hiện cuối cùng trong mảng, -1 nếu không tồn tại. n là số phần tử của mảng arr, low là chỉ số phần tử đầu tiên của mảng, high là chỉ số phần tử cuối cùng của mảng.

```
int last(int arr[], int low, int high, int x, int n)
{
 if (high >= low)
 {
  int mid = (low + high)/2;
  if( ( mid == n-1 || x < ___ ) && arr[mid] == x )
    return mid;
  else if(x < arr[mid])
    return last(arr, low, ___ , x, n);
  else
    return last(arr, ___ , high, x, n);
 }
```

```
  return -1;
}
```

d) Hoàn thành hàm trả về số lần xuất hiện của x, -1 nếu không tồn tại. n là số phần tử của mảng arr.

```
int count(int arr[], int x, int n)
{
  int i, j;

  i = first(arr, 0, n-1, x, n);
  if(i == -1)
    return -1;

  j = last(arr, i, n-1, x, n);
  return j-i+1;
}
```

e) Hoàn thiện hàm main kiểm tra kết quả giải thuật

```
int main()
{
  int k[] = {1, 2, 2, 2, 3, 4, 5};
  int x =  2;
  int n = sizeof(k)/sizeof(k[0]);
  int c = count(k, x, n);

  printf("\nMang da cho: ");
  print_array(k, 0, ___ );
  printf("\n%Phan tu gia tri %d xuat hien %d lan", ___ , ___ );

  return 0;
}
```