BÀI TẬP THỰC HÀNH 01: ÔN TẬP

1 Con trỏ

Sử dụng kĩ thuật con trỏ để hoàn thiện các hàm sau:

1. Viết hàm nhập vào một mảng số nguyên gồm n phần tử với a là con trỏ trỏ tới vùng nhớ của mảng vừa nhập:
   - **void inputArray(int* &a, int &n);**
2. Viết hàm hủy cấp phát động cho mảng:
   - **void dellocateArray(int* &a);**
3. Viết hàm in ra màn hình các giá trị trong mảng:
   - **void printArray(int* a, int n);**
4. Viết hàm tìm giá trị nhỏ nhất trong mảng:
   - **int findMin(int* a, int n);**
5. Viết hàm tìm phần tử có trị tuyệt đối lớn nhất trong mảng:
   - **int findMaxModulus(int* a, int n);**
6. Viết hàm kiểm tra xem mảng có tăng dần hay không:
   - **bool isAscending(int* a, int n);**
7. Viết hàm tính tổng các phần tử trong mảng:
   - **int sumofArray(int* a, int n);**
8. Viết hàm đếm số lượng số nguyên tố trong mảng:
   - **int countPrime(int* a, int n);**
9. Viết hàm đảo ngược mảng mà không dùng mảng phụ:
   - **void reverseArray(int* &a, int n);**

   Từ câu 10. đến 13. yêu cầu tìm kiếm vị trí của giá trị `key` cho trước. Trả về vị trí đầu tiên tìm được. Nếu không tìm được trả về -1.
10. Tìm kiếm tuần tự:
    - **int LinearSearch(int* a, int n, int key);**
11. Tìm kiếm tuần tự (sử dụng phương pháp lính canh):
    - **int sentinelLinearSearch(int* a, int n, int key);**
12. Tìm kiếm nhị phân:
    - **int BinarySearch(int* a, int n, int key);**
13. Tìm kiếm nhị phân (sự dụng đệ quy):
    - **int recursiveBinarySearch(int* a, int left, int right, int key);**

2 Đệ quy

Sử dụng kỹ thuật Đệ quy để giải quyết các yêu cầu sau:

1. Viết hàm tính tổng bình phương các số tự nhiên nhỏ hơn hoặc bằng n: $S = 1^2 + 2^2 + ... + n^2$.
   - **int sumOfSquares(int n);**
2. Viết hàm tìm ước chung lớn nhất của 2 số nguyên a, b:
   - **int gcd(int a, int b);**
3. Số Fibonacci thứ n được tính như sau: $F(n) = F(n-1) + F(n-2)$. Viết hàm tính số Fibonacci thứ n.
   - **int fib(int n);**

3 Danh sách liên kết

Cho một danh sách liên kết đơn được định nghĩa như sau:

```
struct NODE{
    int key;
    NODE* pNext;
};
```

```
struct List{
    NODE* pHead;
    NODE* pTail;
};
```

Sinh viên viết hàm thực hiện các yêu cầu sau:

1. Khởi tạo danh sách rỗng với kiểu dữ liệu `List`:
   - **List* createList();**
2. Khởi tạo một `NODE` từ một số nguyên cho trước:
   - **NODE* createNode(int data);**
3. Chèn một số nguyên vào đầu một `List` cho trước:
   - **bool addHead(List* &L, int Data);**
4. Chèn một số nguyên vào cuối một `List` cho trước:
   - **bool addTail(List* &L, int Data);**
5. Xóa `NODE` đầu tiên của một `List` cho trước:
   - **void removeHead(List* &L);**
6. Xóa `NODE` cuối cùng của một `List` cho trước:
   - **void removeTail(List* &L);**
7. Xóa tất cả các `NODE` của một `List` cho trước:
   - **void removeAll(List* &L);**
8. In tất cả phần tử của một `List` cho trước:
   - **void printList(List* L);**
9. Đếm số lượng phần tử của một `List` cho trước:
   - **int countElements(List* L);**
10. Đảo một `List` cho trước *(tạo ra một* `List` *mới)*:
    - **List* reverseList(List* L);**
11. Xóa tất cả các phần từ trùng của một `List` cho trước:
    - **void removeDuplicate(List* &L);**
12. Xóa giá trị `key` khỏi một `List` cho trước:
    - **bool removeElement(List* &L, int key);**

4 Quy định nộp bài

- Sinh viên nộp bài dưới dạng MSSV.rar(.zip).
- Mỗi phần cần được đặt trong thư mục riêng. Tất cả nằm trong thư mục MSSV (Lưu ý: chỉ nộp file .h và .cpp).
- Các bài làm giống nhau sẽ bị 0 điểm môn học.