ĐỀ THI GIỮA KÌ 21CTT3

Thời gian làm bài: 60 phút

Dữ liệu sử dụng trong đề bài là dữ liệu từ điển Anh - Việt về thú vật dành cho trẻ em có nội dung như sau:

```
5
Piglet
('pɪglət)
Lợn con
1
Plaice
/pleɪs/
Cá bơn
5
Polar bear
('pəʊlə beə)
Con gấu Bắc cực
2
Porcupine
('pɔ:kjʊpaɪn)
Con nhím.
2
Porcupine
('pɔ:kjʊpaɪn)
Con nhím
2
Praying mantis
('preɪɪŋ 'mæntɪs)
Bọ ngựa
```

Cho các định nghĩa struct sau:

```
struct Animal
{
    string en;
    string vn;
    string ph;
    int loai;
};
```

```
struct Node
{
    Animal data;
    Node* pnext;
};
```

```
struct LinkedList
{
    Node* pHead;
    Node* pTail;
};
```

Thực hiện các yêu cầu sau:

Câu 1 (4 điểm)

1. (2 điểm) Đọc dữ liệu động vật lưu vào danh sách liên kết

- `LinkedList* readAnimals(string filename);`
- Input: `filename` - tệp tin dữ liệu *"data.txt"*
- Output: Danh sách liên kết `LinkedList`

2. (2 điểm) Trong dữ liệu có chứa một số động vật bị trùng lặp, hãy chuẩn hóa bằng cách xóa những node bị trùng lặp trong danh sách liên kết, giữ node đầu tiên xuất hiện.

- `void removeDuplicate(LinkedList* list);`

Câu 2 (3 điểm)

Chương trình sau thực hiện tìm tất cả các hướng đi trong một ma trận 2 chiều MxN bắt đầu tại vị trí (i,j) cho trước và kết thúc tại vị trí cuối cùng của ma trận (M-1, N-1). Biết rằng đường đi chỉ có thể đi xuống, đi qua phải hoặc đi theo đường chéo (hướng xuống).

```
void printPaths(int** matrix, vector<int> &route, int len_route, int i, int j, int M, int N)
{
    // MxN matrix
    if(M == 0 || N == 0)
    {
        return;
    }

    // if the last cell is reached
    if(i == M-1 && j == N-1)
    {
        // print the route
        for (int k = 0; k < len_route; k++)
        {
            cout << route[k] << " ";
        }
        cout << matrix[i][j] << endl;
        return;
    }

    // add the current cell to route
    route.push_back(matrix[i][j]);
    len_route += 1;

    // move down
    if (i + 1 < M)
    {
        printPaths(matrix, route, len_route, i+1, j, M, N);
    }

    // move right
    if (j + 1 < N)
    {
        printPaths(matrix, route, len_route, i, j+1, M, N);
    }

    // move diagonally
    if (i + 1 < M && j + 1 < N)
    {
        printPaths(matrix, route, len_route, i+1, j+1, M, N);
    }
```

```
    // backtrack
    route.pop_back();
}

void Bai02()
{
    int M = 3, N=3;
    int** matrix = new int*[M];

    for(int i =0; i < M; i++)
    {
        matrix[i] = new int[N];
    }

    for (int i = 0; i < M*N; i++)
    {
        matrix[i/N][i%N] = i;
    }

    vector<int> route;
    int len_route = 0;
    int i = 0, j = 0;

    // Goi ham printPaths
    printPaths(matrix, route, len_route, i, j, M, N); // Ban co the thay doi dong nay neu co thay doi tham so
        cua ham

    // In ra so phep gan v so phep so sanh
    // CODE HERE
}
```

Sinh viên điều chỉnh mã nguồn của hàm `PrintPaths` để đếm số phép gán và số phép so sánh của hàm `PrintPaths`. Sau đó in ra số phép gán và số phép so sánh trong `Bai02()`
Lưu ý: Bỏ qua chi phí của các phương thức `pop_back()` và `push_back()`

Câu 3 (3 điểm)

(1.5 điểm) Viết chương trình xuất ra phần tử nhỏ thứ k của một mảng số nguyên cho trước.

```
int kthSmallest(int* arr, int n, k);
```

Hãy đảm bảo chi phí thuật toán của bạn:

1. (1 điểm) Về mặt thời gian có độ phức tạp nhỏ hơn O(n^2)
2. (0.5 điểm) Về mặt không gian là có chi phí là O(1)