TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN, ĐHQG-HCM

ĐỀ THI CUỐI KỲ

Học kỳ 1 – Năm học 2025 - 2026

Tên học phần: **Cơ Sở Lập Trình** Đề: **02**

Thời gian làm bài: **100 phút** Ngày thi:

*Ghi chú: Sinh viên **được phép sử dụng** tài liệu khi làm bài*

Câu 1

1.1. Vẽ hình bằng dấu "*"

Viết hàm:

```
void hollowPyramid(int h);
```

Vẽ hình kim tự tháp rỗng (hollow pyramid) bằng dấu * với chiều cao **h ≥ 3**.

Ví dụ với **h = 6**:

```
     *
    * *
   *   *
  *     *
 *       *
***********
```

1.2. Xác định output của chương trình sau và giải thích

```
#include <iostream>
using namespace std;

void process(int *p, int a[], int n) {
    *p = a[1] + a[2];
    a[2] = *p + a[0];
    p = &a[a[0]];
    *p = a[2] - a[1];
}

int main() {
    int a[5] = {2, 5, 1, 4, 3};
    int *q = &a[3];

    process(q, a, 5);

    cout << a[0] << " " << a[1] << " " << a[2] << " "
         << a[3] << " " << a[4] << endl;

    return 0;
}
```

Yêu cầu:

- Viết output
- Giải thích chi tiết quá trình thay đổi mảng & con trỏ

Câu 2

Tìm mảng con có tổng lớn thứ **k**. Một **mảng con** (subarray) là mảng được tạo từ **các phần tử liên tiếp** trong mảng ban đầu.

Cho mảng số nguyên **a** gồm **n** phần tử và số nguyên **k**.

Viết hàm:

```
int* kthLargestSubarray(int a[], int n, int k, int &subSize);
```

Yêu cầu:

- Trong tất cả mảng con của **a**. Tìm một mảng con có tổng lớn thứ **k**
- Hàm trả về con trỏ quản lý mảng con kết quả
- **subSize** là kích thước của mảng con trả về
- Nếu **k** lớn hơn số lượng mảng con → **subSize = 0**, trả về **nullptr**

Ví dụ:

```
a = [1, -2, 3]
Các mảng con:
[1]             → 1
[1, -2]         → -1
[1, -2, 3]      → 2
[-2]            → -2
[-2, 3]         → 1
[3]             → 3
```

Nếu k = 2 → tổng lớn thứ 2 là 2 → mảng con trả về là **[1, -2, 3]**.

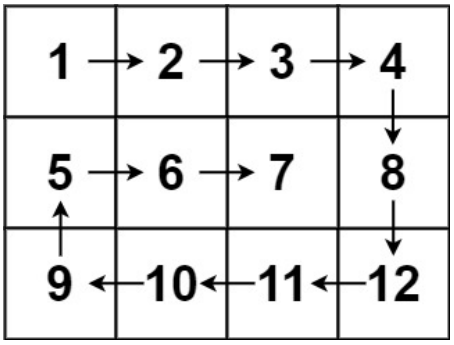
Câu 3

Cho ma trận **m** × **n** gồm số nguyên dương.

Tìm ma trận có diện tích lớn nhất sao cho:

1. Là hình chữ nhật liên tiếp trong ma trận gốc
2. Khi duyệt theo **xoắn ốc (spiral order)** các giá trị thu được **tăng dần**

Minh họa cách duyệt xoắn ốc



Viết hàm:

```
int** largestIncreasingSpiral(int **a, int m, int n, int &rows, int &cols);
```

Nếu không có ma trận con hợp lệ → trả về **nullptr** và đặt **rows = cols = 0**.

Câu 4

Cho danh sách liên kết đơn và giá trị **x**.

Phân chia danh sách sao cho:

- Các node có giá trị < **x** đứng trước
- Các node có giá trị ≥ **x** đứng sau
- **Thứ tự tương đối** giữa các phần tử trong mỗi phần được giữ nguyên

Ví dụ:

```
Input:
1 → 4 → 3 → 2 → 5 → 2    (x = 3)

Output:
1 → 2 → 2 → 4 → 3 → 5
```

Viết hàm:

```
Node* partitionList(Node* head, int x);
```

Câu 5

Thống kê bill bán lẻ của một quán cafe được lưu trong file **bill.txt**, mỗi dòng:

```
Tên món | số lượng | đơn giá
```

Ví dụ:

```
Capuchino|2|45000
MatchaLatte|1|40000
BacXiu|3|30000
TraSua|2|35000
Capuchino|1|45000
```

5.1. Tính tổng doanh thu

Viết hàm:

```
int totalRevenue(const char* filename);
```

5.2. Tính tổng số lượng từng món

```
Xuất dạng:
Capuchino: 3 ly
MatchaLatte: 1 ly
BacXiu: 3 ly
...
```

5.3. Tìm món doanh thu cao nhất và in ra màn hình

Ví dụ:

```
Món doanh thu cao nhất: Capuchino (135000 VND)
```

5.4. Liệt kê các món bán ≥ 2 ly

Format:

```
Tên|Số lượng|Đơn giá|Doanh thu
```

5.5. Sắp xếp danh sách món theo doanh thu giảm dần và ghi vào file "output.txt"

Format:

```
Tên|Số lượng|Đơn giá|Doanh thu
```

Câu 6:

Cho dữ liệu tương tự câu 5 nhưng được lưu dưới dạng nhị phân "**bill.bin**".

6.1 Viết hàm đọc file.

```
Node* readFile(const char *filename);
```

Khai báo struct **Node**, sau đó đọc file lưu vào danh sách liên kết.

6.2 Chuẩn hóa tên món.

Hiện tại tên món đang được đặt theo style **Camel**. Ví dụ từ **ca phe muoi** sẽ được viết là **CaPheMuoi**. Sửa thông tin file **bill.bin** đổi tên các món từ dạng **Camel** về dạng thường. Ví dụ: **CaPheMuoi → ca phe muoi.**

--- HẾT ---