Bài tập tuần 5

1.Tìm hiểu về các hàm cấp phát bộ nhớ động của C

- **malloc():**

- Từ malloc là đại diện cho cụm từ memory allocation (dịch: cấp phát bộ nhớ).

- Hàm malloc() thực hiện cấp phát bộ nhớ bằng cách chỉ định số byte cần cấp phát. Hàm này trả về con trỏ kiểu void cho phép chúng ta có thể ép kiểu về bất cứ kiểu dữ liệu nào.

- Cú pháp của hàm malloc():
  ptr = (castType*) malloc(size);

- Ví dụ:
  ptr = (int*) malloc(100 * sizeof(int));

Ví dụ trên thực hiện cấp phát cho việc lưu trữ 100 số nguyên. Giả sử sizeof int là 4, khi đó lệnh dưới đây thực hiện cấp phát 400 bytes. Khi đó, con trỏ ptr sẽ có giá trị là địa chỉ của byte dữ liệu đầu tiên trong khối bộ nhớ vừa cấp phát. Trong trường hợp không thể cấp phát bộ nhớ, nó sẽ trả về một con trỏ NULL.

- **calloc():**

- Từ calloc đại diện cho cụm từ contiguous allocation (dịch: cấp phát liên tục).

- Hàm malloc() khi cấp phát bộ nhớ thì vùng nhớ cấp phát đó không được khởi tạo giá trị ban đầu. Trong khi đó, hàm calloc() thực hiện cấp phát bộ nhớ và khởi tạo tất cả các ô nhớ có giá trị bằng 0.

- Hàm calloc() nhận vào 2 tham số là số ô nhớ muốn khởi tạo và kích thước của 1 ô nhớ.

- Cú pháp của hàm calloc():
  ptr = (castType*)calloc(n, size);

- Ví dụ:
  ptr = (int*) calloc(100, sizeof(int));

Trong ví dụ trên, hàm calloc() thực hiện cấp phát 100 ô nhớ liên tiếp và mỗi ô nhớ có kích thước là số byte của kiểu int. Hàm này cũng trả về con trỏ chứa giá trị là địa chỉ của byte đầu tiên trong khối bộ nhớ vừa cấp phát.

- **free():**

- Việc cấp phát bộ nhớ động trong C dù sử dụng malloc() hay calloc() thì chúng cũng đều không thể tự giải phóng bộ nhớ -> cần sử dụng hàm free() để giải phóng vùng nhớ.

- Cú pháp:
  free(ptr); // ptr là con trỏ

- Lệnh này sẽ giải phóng vùng nhớ mà con trỏ ptr đã được cấp phát. Giải phóng ở đây có nghĩa là trả lại vùng nhớ đó cho hệ điều hành và hệ điều hành có thể sử dụng vùng nhớ đó vào việc khác nếu cần.

- Nếu không giải phóng nó thì nó sẽ tồn tại cho tới khi chương trình kết thúc. Điều này sẽ rất nguy hiểm nếu chương trình liên tục cấp phát các vùng nhớ mới và sẽ gây ra hiện tượng tràn bộ nhớ.

- **realloc():**

- Nếu việc cấp phát bộ nhớ động không đủ hoặc cần nhiều hơn mức đã cấp phát, chúng ta có thể thay đổi kích thước của bộ nhớ đã được cấp phát trước đó bằng cách sử dụng hàm realloc().

- Cú pháp của realloc():
  ptr = realloc(ptr, n);

- Hàm này thực hiện cấp phát vùng nhớ mới cho con trỏ ptr. Vùng nhớ mới đó sẽ có kích thước mới là n bytes.

- Hàm này cũng trả về con trỏ chứa giá trị là địa chỉ của byte đầu tiên trong vùng nhớ mới. Hàm này sẽ cố gắng mở rộng số ô nhớ ra phía sau nếu có thể để giữ nguyên giá trị của con trỏ ban đầu. Trong trường hợp phải đổi sang một vùng nhớ khác, hàm realloc() cũng sẽ mang theo giá trị đã có ở vùng nhớ cũ sang vùng nhớ mới và giải phóng luôn vùng nhớ cũ. Trong trường hợp không thể, nó sẽ trả về con trỏ NULL.

- Phân biệt malloc() và calloc():

malloc()	**calloc()**
- malloc = memory allocation	- calloc = contiguous allocation
- Nhận 1 tham số truyền vào là số byte của vùng nhớ cần cấp phát	- Nhận 2 tham số truyền vào là số block và kích thước mỗi block (byte)
- Cú pháp:	- Cú pháp:

ptr = (castType*) malloc(size);	**ptr = (castType*)calloc(n, size);**
- Hàm trả về con trỏ trỏ tới vùng nhớ nếu cấp phát thành công, trả về NULL nếu cấp phát không thành công	- Hàm trả về con trỏ trỏ tới vùng nhớ được cấp phát và vùng nhớ được khởi tạo bằng giá trị 0, trả về NULL nếu cấp phát không thành công
- Hàm malloc() nhanh hơn hàm calloc()	- Hàm calloc() tốn thêm thời gian khởi tạo bộ nhớ nên chậm hơn nhưng không đáng kể

2.Xây dựng ct nhân ma trận dùng bộ nhớ động với ** theo ví dụ cộng ma trận trong slide cũ
3.Xây dựng các hàm cấp phát và giải phóng bộ nhớ động cho mảng 3 chiều dùng *** và ****
4.bài toán qlsv :
- Dùng bộ nhớ động để tiết kiệm tối đa bộ nhớ (mảng zíc zắc và dư thừa tối đa 50 con trỏ)
- Sắp xếp kiểu VN
+ Dùng hàm đảo tên để chỉ dùng hàm strcmp 1 lần là sắp xếp được đồng thời chỉ ra các vấn đề gặp phải khi làm theo cách này
+ Dùng hàm vị trí tên để lấy ra tên sv thứ i so sánh với tên sv thứ j, nếu không theo thứ tự thì hoàn vị, nếu = nhau thì so sánh tiếp cả hovaten của sv thứ i với sv thứ j, nếu không theo thứ tự thì hoán vị (dùng hàm strcmp 3 lần)
+ Kết hợp 2 phương pháp trên để chỉ dùng strcmp 1 lần và không gặp các vấn đề của phương pháp 1