Câu hỏi 1

Đúng

Đạt điểm 10,00 trên 10,00

Yêu cầu

Viết hàm `int* getPointerToTen()` trả về một con trỏ đến số nguyên có giá trị bằng 10.

Input

- Không có đầu vào

Output

- Khai báo một con trỏ kiểu nguyên trong hàm
- Cấp phát bộ nhớ của một số nguyên cho con trỏ đó
- Gán giá trị 10 cho vùng bộ nhớ đó và trả về con trỏ đã khai báo.

Gợi ý

1. Khai báo con trỏ đến số nguyên bằng lệnh `int* pInt;`
2. Xin cấp phát bộ nhớ chứa số nguyên bằng lệnh `pInt = new int;`
3. Gán giá trị cho vùng nhớ vừa cấp phát: `*pInt = 10;`
4. Trả về con trỏ đến vùng nhớ vừa xin cấp phát: `return pInt;`

Lưu ý

1. Khai báo con trỏ bằng cú pháp `<kiểu> * <tên biến con trỏ>;`
2. Có thể xin cấp phát ngay khi khai báo: `int* pInt = new int;`
3. Truy xuất đến vùng nhớ trỏ đến bởi con trỏ bằng toán tử `*`
4. Trong bài này, ta có thể trả về giá trị của một biến trong hàm (ở đây là biến con trỏ). Tuy nhiên, không nên trả về con trỏ đến một biến được khai báo trong hàm vì biến này sẽ bị xóa khỏi bộ nhớ khi hàm kết thúc. Ví dụ, không được làm bài này như sau

   ~~`int x = 10;`~~

   ~~`return &x;`~~

Answer: (penalty regime: 0 %)

```
int* getPointerToTen() {
  int *p = new int(10);
  return p;
}
```

Passed all tests! ✓

Đúng

Marks for this submission: 10,00/10,00.

Câu hỏi 2

Đúng

Đạt điểm 10,00 trên 10,00

Yêu cầu

Viết hàm `void multiply (int* n, int k)` thực hiện phép nhân giá trị do con trỏ n trỏ tới lên k lần.

Input

- Đầu vào thứ nhất là con trỏ đến vùng nhớ chứa một số nguyên n
- Đầu vào thứ hai là tỉ số k

Output

- Hàm không yêu cầu trả về giá trị
- Nhưng khi hàm kết thúc, giá trị do con trỏ n trỏ đến được nhân lên k lần.

Gợi ý

1. Khi truyền một con trỏ vào hàm, mọi thay đổi với biến con trỏ sẽ làm thay đổi giá trị của biến tương ứng bên ngoài hàm.
2. Cách truyền tham số này gọi là truyền tham số bằng con trỏ.
3. Bài này chỉ cần nhân giá trị do n trỏ đến lên k lần bằng lệnh `*n = (*n) * k;`

Lưu ý

1. Việc sử dụng con trỏ rất linh hoạt và hiệu quả, tăng tốc độ chương trình vì truy xuất trực tiếp đến bộ nhớ của các biến
2. Tuy nhiên, khi sử dụng con trỏ cần kiểm tra kỹ con trỏ có hợp lệ hay không bằng điều kiện `n != nullptr` hoặc `n != NULL`

For example:

Input	Result
2 3	6

Answer: (penalty regime: 0 %)

```
void multiply(int *n, int k) {
  *n *= k;
}
```

	Input	Expected	Got	
✓	2 3	6	6	✓
✓	-122 45	-5490	-5490	✓
✓	123 11	1353	1353	✓

Passed all tests! ✓

Đúng

Marks for this submission: 10,00/10,00.

Câu hỏi 3

Đúng

Đạt điểm 10,00 trên 10,00

Yêu cầu:

Việc hoán đổi giá trị của hai biến được sử dụng trong nhiều bài toán, chẳng hạn như bài toán sắp xếp. Viết hàm `void swap(int* a, int* b)` thực hiện việc hoán đổi giá trị của hai biến a và b.

Input:

- Con trỏ lưu địa chỉ của biến thứ nhất và con trỏ lưu địa chỉ của biến thứ hai.

Output:

- Hoán đổi giá trị trong 2 địa chỉ đầu vào.

Gợi ý:

1. Khởi tạo biến tạm.
2. Gán giá trị biến tạm bằng giá trị ở ô địa chỉ con trỏ thứ nhất trỏ tới.
3. Gán giá trị ở ô địa chỉ con trỏ thứ nhất trỏ tới bằng giá trị ở ô địa chỉ con trỏ thứ hai trỏ tới.
4. Gán giá trị ở ôn địa chỉ con trỏ thứ hai trỏ tới bằng biến tạm.

For example:

Input	Result
1 2	2 1

Answer: (penalty regime: 0 %)

```
// new technique :>
void swap(int *a, int *b) {
  *a = *a ^ *b;
  *b = *a ^ *b;
  *a = *a ^ *b;
}
```

	Input	Expected	Got	
✓	1 2	2 1	2 1	✓
✓	3 4	4 3	4 3	✓
✓	123456789 987654321	987654321 123456789	987654321 123456789	✓
✓	100 1000	1000 100	1000 100	✓
✓	444 555	555 444	555 444	✓

Passed all tests! ✓

Đúng

Marks for this submission: 10,00/10,00.

Câu hỏi **4**

Đúng

Đạt điểm 10,00 trên 10,00

Yêu cầu:

Viết hàm `int* getPointerToArray(int n)`. Hàm này khai báo một con trỏ kiểu nguyên, cấp phát bộ nhớ động cho con trỏ đó n phần tử kiểu nguyên và sau đó gán giá trị cho n phần tử đó các số được nhập từ bàn phím. Hàm trả về con trỏ được khai báo.

Input:

- Số nguyên $n, 0 < n \leq 1000$

Output:

- Con trỏ trỏ tới vùng nhớ lưu trữ n giá trị đã nhập từ bàn phím

Gợi ý:

1. Khai báo con trỏ: `int *arr;`
2. Cấp phát bộ nhớ động cho con trỏ đã khai báo với kích thước n: `arr = new int [n];`
3. Dùng vòng for để nhập dữ liệu n số nguyên từ bàn phím.
4. Trả về con trỏ

Lưu ý:

1. SINH VIÊN KHÔNG ĐƯỢC KHAI BÁO MẢNG THEO CÁCH NÀY `int array[n];`, phải sử dụng cấp phát động.

For example:

Input	Result
3 -5021 4497 -9846	-5021 4497 -9846

Answer: (penalty regime: 0 %)

```
int* getPointerToArray(int n) {
  int* a = new int[n];
  for (int i = 0; i < n; ++i) {
    cin >> a[i];
  }
  return a;
}
```

	Input	Expected	Got	
✓	3 -5021 4497 -9846	-5021 4497 -9846	-5021 4497 -9846	✓
✓	16 8948 -301 6740 4125 6514 7332 7390 4547 5507 -6894 -9818 -1065 -4962 7746 5690 -5259	8948 -301 6740 4125 6514 7332 7390 4547 5507 -6894 -9818 -1065 -4962 7746 5690 -5259	8948 -301 6740 4125 6514 7332 7390 4547 5507 -6894 -9818 -1065 -4962 7746 5690 -5259	✓
✓	80 -9849 5239 -627 5864 -5636 -578 753 9474 -1362 9909 1007 -1055 1422 1988 8792 2458 -8720 -697 4733 -234 -6538 57 -1116 9797 -4540 -5924 1824 7688 -2977 6631 -9555 -7784 1356 -3256 -2966 6415 4605 5644 -514 2842 3686 2119 -1173 -3325 -9371 5325 -3710 4362 5603 -4205 -4336 344 -2294 -6984 -9429 607 9374 -4523 -311 9268 7430 -2751 8830 -3061 2166 5946 6150 7410 7982 -9216 3384 -2 4582 1105 -2627 -1705 8867 -7383 6971 -500	-9849 5239 -627 5864 -5636 -578 753 9474 -1362 9909 1007 -1055 1422 1988 8792 2458 -8720 -697 4733 -234 -6538 57 -1116 9797 -4540 -5924 1824 7688 -2977 6631 -9555 -7784 1356 -3256 -2966 6415 4605 5644 -514 2842 3686 2119 -1173 -3325 -9371 5325 -3710 4362 5603 -4205 -4336 344 -2294 -6984 -9429 607 9374 -4523 -311 9268 7430 -2751 8830 -3061 2166 5946 6150 7410 7982 -9216 3384 -2 4582 1105 -2627 -1705 8867 -7383 6971 -500	-9849 5239 -627 5864 -5636 -578 753 9474 -1362 9909 1007 -1055 1422 1988 8792 2458 -8720 -697 4733 -234 -6538 57 -1116 9797 -4540 -5924 1824 7688 -2977 6631 -9555 -7784 1356 -3256 -2966 6415 4605 5644 -514 2842 3686 2119 -1173 -3325 -9371 5325 -3710 4362 5603 -4205 -4336 344 -2294 -6984 -9429 607 9374 -4523 -311 9268 7430 -2751 8830 -3061 2166 5946 6150 7410 7982 -9216 3384 -2 4582 1105 -2627 -1705 8867 -7383 6971 -500	✓
✓	90 3172 3094 -5447 9503 849 -6219 -7943 4223 5200 1233 3201 2972 -6153 6332 2566 8931 -9307 -9509 -7624 -6024 -9759 7715 6451 -8462 5536 -1893 -6058 -8139 4732 -3097 -4412 1962 5915 -7670 3659 -6502 6254 7283 5071 -2977 9858 -9310 -9035 8775 1973 -1663 5367 4450 1755 8973 1771 -725 -918 -2010 9319 8946 -239 2201 -7783 -8594 1857 1687 6041 -1702 6823 -2792 -388 -3566 7863 9896 -8312 3893 3540 -5050 -8057 4922 7796 4268 -7522 2875 -7151 3936 863 6541 8539 -6609 7506 3460 7385 446	3172 3094 -5447 9503 849 -6219 -7943 4223 5200 1233 3201 2972 -6153 6332 2566 8931 -9307 -9509 -7624 -6024 -9759 7715 6451 -8462 5536 -1893 -6058 -8139 4732 -3097 -4412 1962 5915 -7670 3659 -6502 6254 7283 5071 -2977 9858 -9310 -9035 8775 1973 -1663 5367 4450 1755 8973 1771 -725 -918 -2010 9319 8946 -239 2201 -7783 -8594 1857 1687 6041 -1702 6823 -2792 -388 -3566 7863 9896 -8312 3893 3540 -5050 -8057 4922 7796 4268 -7522 2875 -7151 3936 863 6541 8539 -6609 7506 3460 7385 446	3172 3094 -5447 9503 849 -6219 -7943 4223 5200 1233 3201 2972 -6153 6332 2566 8931 -9307 -9509 -7624 -6024 -9759 7715 6451 -8462 5536 -1893 -6058 -8139 4732 -3097 -4412 1962 5915 -7670 3659 -6502 6254 7283 5071 -2977 9858 -9310 -9035 8775 1973 -1663 5367 4450 1755 8973 1771 -725 -918 -2010 9319 8946 -239 2201 -7783 -8594 1857 1687 6041 -1702 6823 -2792 -388 -3566 7863 9896 -8312 3893 3540 -5050 -8057 4922 7796 4268 -7522 2875 -7151 3936 863 6541 8539 -6609 7506 3460 7385 446	✓
✓	24 2774 -567 -2701 -9067 9837 418 -7935 4937 2756 -517 3981 5548 -6575 254 2385 -8120 -7676 -149 -2200 -3833 520 4328 1840 -4811	2774 -567 -2701 -9067 9837 418 -7935 4937 2756 -517 3981 5548 -6575 254 2385 -8120 -7676 -149 -2200 -3833 520 4328 1840 -4811	2774 -567 -2701 -9067 9837 418 -7935 4937 2756 -517 3981 5548 -6575 254 2385 -8120 -7676 -149 -2200 -3833 520 4328 1840 -4811	✓

Passed all tests! ✓

Đúng

Marks for this submission: 10,00/10,00.

Câu hỏi **5**

Đúng

Đạt điểm 10,00 trên 10,00

Yêu cầu:

Viết hàm tính bình phương của một số thực.

Hàm `double* getSquare (double number)` nhận đầu vào là một số thực $number$ và trả về con trỏ kiểu $double$ chứa giá trị bình phương của số thực đã cho.

Input:

- Một số thực `number`

Output:

- Output của hàm là con trỏ kiểu double chứa giá trị bình phương của số đã cho
- Kết quả làm tròn đến chữ số thập phân thứ hai.

Gợi ý:

1. Cấp phát 1 con trỏ kiểu double ở trong hàm cần viết `double* square = new double;`
2. Thao tác với giá trị của biến con trỏ thông qua phép toán * , `*square = ...`

Lưu ý:

1. Trong bài này ta vẫn dùng cách truyền tham số bằng giá trị
2. Bạn không cần phải làm tròn trong hàm, trong hàm main có sẵn đã làm tròn kết quả trả về của hàm.

Answer: (penalty regime: 0 %)

```
double* getSquare(double number) {
  double* p = new double(number * number);
  return p;
}
```

	Input	Expected	Got	
✓	2	4.00	4.00	✓
✓	-1	1.00	1.00	✓
✓	20.13	405.22	405.22	✓
✓	109.242341	11933.89	11933.89	✓
✓	743.2130123213	552365.58	552365.58	✓

Passed all tests! ✓

Đúng

Marks for this submission: 10,00/10,00.

Câu hỏi 6

Đúng

Đạt điểm 10,00 trên 10,00

Yêu cầu:

Viết hàm `int* merge(int* firstArr, int lenArr1, int* secondArr, int lenArr2)` thực hiện việc nối hai mảng số nguyên đã sắp xếp với độ dài biết trước thành một mảng số nguyên duy nhất và thứ tự sắp xếp không đổi.

Input:

- Hàm nhận đầu vào là hai mảng số nguyên đã sắp xếp $firstArr$ và $secondArr$, với độ dài mảng lần lượt là $lenArr1$ và $lenArr2$.

Output:

- Hàm trả về mảng số nguyên là kết quả của việc nối hai mảng đầu vào thành mảng duy nhất và các phần tử trong mảng cũng được sắp xếp theo thứ tự không đổi (tăng dần hoặc giảm dần) như các mảng đầu vào.

Gợi ý:

1. Cấp phát động 1 mảng số nguyên $mergeArr$ có kích thước bằng $lenArr1 + lenArr2$.
2. Kiểm tra xem các mảng đầu vào là sắp xếp tăng dần hay giảm dần.
3. Có 2 cách chính để giải quyết bài toán này.
    - Cách 1: copy các phần tử của 2 mảng sang mảng $mergeArr$ sau đó dùng thuật toán sắp xếp để sắp xếp mảng đó theo chiều tăng dần hay giảm dần đã xác định ở bước trước. Cách này đơn giản, dễ thực hiện nhưng thời gian chạy chương trình sẽ lâu.
    - Cách 2: Giả sử mảng đang sắp xếp tăng dần, trường hợp mảng sắp xếp giảm dần làm tương tự.
        - Với mỗi 1 mảng sẽ có 1 chỉ số chạy, ví dụ $firstIndex$ cho $firstArr$ và $secondIndex$ cho $secondArr$.
        - Với mỗi bước lặp ta sẽ so sánh các phần tử ở 2 mảng, nếu phần từ đang xét ở mảng $firstArr$ nhỏ hơn hoặc bằng phần tử đang xét ở mảng $secondArr$ thì ta thêm phần tử ở mảng $firstArr$ vào mảng $mergeArr$ ngược lại thì thêm phần tử ở mảng $secondArr$
        - Khi chạy xong ở bước trên, mảng nào còn thừa phần tử thì ta sẽ thêm vào cuối của mảng $mergeArr$

Lưu ý:

1. Các mảng đầu vào có thể sắp xếp tăng dần hoặc giảm dần.
2. Tham khảo code mẫu dưới đây cho Cách 2 trường hợp mảng sắp xếp tăng dần

```
int* merge(int* firstArr, int lenArr1, int* secondArr, int lenArr2){
    int *mergeArr = ...;  // Tạo mảng mergeArr và cấp phát
    int firstIndex = 0, secondIndex = 0, mergeIndex = 0;
    while (firstIndex < lenArr1 && secondIndex < lenArr2){
        if(firstArr[firstIndex] <= secondArr[secondIndex]){
            mergeArr[mergeIndex] = firstArr[firstIndex];
            firstIndex++;
        }else{
            mergeArr[mergeIndex] = secondArr[secondIndex];
            secondIndex++;
        }
        mergeIndex++;
    }
    while(firstIndex < lenArr1){
        // Nếu mảng firstArr còn phần tử thì thêm vào cuối mảng mergeArr
    }

    while(secondIndex < lenArr2){
        // Nếu mảng secondArr còn phần tử thì thêm vào cuối mảng mergeArr
    }
    return mergeArr;
}
```

For example:

Input	Result
8 0 5 5 15 23 25 33 61 9 12 25 36 44 45 45 53 65 71	0 5 5 12 15 23 25 25 33 36 44 45 45 53 61 65 71

Answer: (penalty regime: 0 %)

```
int* merge(int* firstArr, int lenArr1, int* secondArr, int lenArr2) {
  int* arr = new int[lenArr1 + lenArr2];
  int i = 0;
  int j = 0;
  int k = 0;
  bool decr = (lenArr1 > 1 && firstArr[0] > firstArr[1]) || (lenArr2 > 1 && secondArr[0] > secondArr[1])
  while (i < lenArr1 && j < lenArr2) {
    arr[k++] = decr ^ (firstArr[i] < secondArr[j])? firstArr[i++] : secondArr[j++];
  }
  while (i < lenArr1) { arr[k++] = firstArr[i++]; }
  while (j < lenArr2) { arr[k++] = secondArr[j++]; }
  return arr;
}
```

	Input	Expected	Got	
✓	8 0 5 5 15 23 25 33 61 9 12 25 36 44 45 45 53 65 71	0 5 5 12 15 23 25 25 33 36 44 45 45 53 61 65 71	0 5 5 12 15 23 25 25 33 36 44 45 45 53 61 65 71	✓
✓	5 46 58 67 74 77 17 10 18 21 22 22 27 28 33 35 37 47 47 53 54 62 77 80	10 18 21 22 22 27 28 33 35 37 46 47 47 53 54 58 62 67 74 77 77 80	10 18 21 22 22 27 28 33 35 37 46 47 47 53 54 58 62 67 74 77 77 80	✓
✓	6 61 55 54 44 20 1 12 67 64 57 38 37 36 36 36 27 5 0 0	67 64 61 57 55 54 44 38 37 36 36 36 27 20 5 1 0 0	67 64 61 57 55 54 44 38 37 36 36 36 27 20 5 1 0 0	✓
✓	50 2043 1998 1930 1927 1886 1877 1836 1775 1735 1686 1680 1601 1552 1473 1428 1369 1309 1251 1237 1227 1175 1158 1145 1122 1120 1029 1001 999 922 897 833 829 763 707 674 495 494 456 455 436 435 417 250 202 188 169 156 130 119 20 41 1997 1984 1914 1914 1909 1829 1731 1718 1672 1668 1607 1398 1325 1297 1264 1150 1106 982 948 935 918 905 768 712 700 522 511 466 464 350 288 278 230 216 165 142 81 52 40 28 7	2043 1998 1997 1984 1930 1927 1914 1914 1909 1886 1877 1836 1829 1775 1735 1731 1718 1686 1680 1672 1668 1607 1601 1552 1473 1428 1398 1369 1325 1309 1297 1264 1251 1237 1227 1175 1158 1150 1145 1122 1120 1106 1029 1001 999 982 948 935 922 918 905 897 833 829 768 763 712 707 700 674 522 511 495 494 466 464 456 455 436 435 417 350 288 278 250 230 216 202 188 169 165 156 142 130 119 81 52 40 28 20 7	2043 1998 1997 1984 1930 1927 1914 1914 1909 1886 1877 1836 1829 1775 1735 1731 1718 1686 1680 1672 1668 1607 1601 1552 1473 1428 1398 1369 1325 1309 1297 1264 1251 1237 1227 1175 1158 1150 1145 1122 1120 1106 1029 1001 999 982 948 935 922 918 905 897 833 829 768 763 712 707 700 674 522 511 495 494 466 464 456 455 436 435 417 350 288 278 250 230 216 202 188 169 165 156 142 130 119 81 52 40 28 20 7	✓
✓	3 65 52 24 34 98 95 93 90 88 82 79 76 75 73 70 67 60 56 55 55 50 39 35 34 32 31 30 30 29 26 23 23 20 16 12 12 4 3	98 95 93 90 88 82 79 76 75 73 70 67 65 60 56 55 55 52 50 39 35 34 32 31 30 30 29 26 24 23 23 20 16 12 12 4 3	98 95 93 90 88 82 79 76 75 73 70 67 65 60 56 55 55 52 50 39 35 34 32 31 30 30 29 26 24 23 23 20 16 12 12 4 3	✓

Passed all tests! ✓

Đúng

Marks for this submission: 10,00/10,00.

Câu hỏi 7

Đúng

Đạt điểm 10,00 trên 10,00

Yêu cầu:

Viết hàm `int** inputMatrix(int nRows, int nCols)` đọc từ bàn phím một ma trận số nguyên có số hàng `nRows` và số cột `nCols`, lưu vào một mảng động hai chiều và trả về con trỏ trỏ đến mảng động này.
Viết hàm `void printMatrix(int** matrix, int nRows, int nCols)`nhận tham số là con trỏ đến mảng động hai chiều `matrix`, số hàng `nRows` và số cột `nCols` của ma trận. Hàm này in ma trận đầu vào ra màn hình, các phần tử trên cùng một hàng cách nhau bởi một dấu cách.

Input:

- Tham số `nRows` và `nCols` lần lượt tương ứng với số hàng và số cột của ma trận.
- Tham số `matrix` (ở hàm thứ 2) là con trỏ đến mảng động hai chiều chứa `nRows` hàng và `nCols` cột.

Output:

- Hàm `inputMatrix` trả về con trỏ trỏ đến một mảng động 2 chiều.
- Hàm `printMatrix` không trả về gì, chỉ in ra màn hình ma trận đầu vào.

Gợi ý:

1. Trong hàm `inputMatrix`, khai báo con trỏ `int** matrix`. Cấp phát bộ nhớ động cho con trỏ đã khai báo với cú pháp sau:

```
int** matrix = new int*[nRows];
for(int i = 0; i < nRows; i++) {
    matrix[i] = new int[nCols];
}
```

   Dùng 2 vòng lặp lồng nhau để đọc từng giá trị vào ma trận từ bàn phím. Sau đó trả về con trỏ `return matrix;`.
2. Trong hàm `printMatrix`, dùng 2 vòng lặp lồng nhau để in ra từng giá trị trong ma trận đầu vào theo cú pháp: `cout << matrix[i][j] << " ";`.

Lưu ý:

1. Con trỏ `matrix` trỏ đến 1 mảng động của các con trỏ, nghĩa là mỗi phần tử `matrix[i]` trong mảng này thực chất cũng là 1 con trỏ trỏ đến 1 mảng int.

For example:

Input	Result
2 3 1 2 3 3 4 5	1 2 3 3 4 5

Answer: (penalty regime: 0 %)

```
int** inputMatrix(int nRows, int nCols) {
  int** a = new int*[nRows];
  for (int i = 0; i < nRows; ++i) {
    a[i] = new int[nCols];
    for (int j = 0; j < nCols; ++j) {
      cin >> a[i][j];
    }
  }
  return a;
}

void printMatrix(int **matrix, int nRows, int nCols) {
  for (int i = 0; i < nRows; ++i) {
    for (int j = 0; j < nCols; ++j) {
      cout << matrix[i][j] << ' ';
    }
```

```
    cout << '\n';
  }
}
```

	Input	Expected	Got	
✓	2 3 1 2 3 3 4 5	1 2 3 3 4 5	1 2 3 3 4 5	✓
✓	3 3 1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	✓
✓	3 2 -1 -2 -3 -4 -5 -6	-1 -2 -3 -4 -5 -6	-1 -2 -3 -4 -5 -6	✓
✓	5 5 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25	✓
✓	1 1 1	1	1	✓

Passed all tests! ✓

Đúng

Marks for this submission: 10,00/10,00.

Câu hỏi 8

Đúng

Đạt điểm 10,00 trên 10,00

Yêu cầu:

Ma trận chuyển vị là một ma trận ở đó các hàng được thay thế bằng các cột, và ngược lại.

Hãy viết hàm `int** transpose(int** matrix, int nRows, int nCols)` nhận đầu vào là một ma trận $matrix$ bất kỳ có kích cỡ $nRows \times nCols$ chứa các giá trị nguyên.

Hàm trả về ma trận chuyển vị của ma trận $matrix$ là ma trận $matrix^T$ có kích thước $nCols \times nRows$ với các hàng được thay thế bởi các cột.

Input:

- Tham số `nRows` và `nCols` lần lượt tương ứng với số hàng và số cột của ma trận.
- Tham số `matrix` là con trỏ đến mảng động hai chiều chứa `nRows` hàng và `nCols` cột.

Output:

- Hàm trả về con trỏ đến một mảng động 2 chiều là ma trận kết quả của phép chuyển vị, có `nCols` hàng và `nRows` cột.

Gợi ý:

1. Khai báo con trỏ `int** transposedMatrix;` và cấp phát bộ nhớ động cho con trỏ đã khai báo với `nCols` hàng và `nRows` cột.
2. Dùng 2 vòng lặp lồng nhau duyệt từng phần tử của ma trận `matrix`, gán phần tử hàng i cột j của matrix cho hàng j cột i của `transposedMatrix` như sau:
   `transposedMatrix[j][i] = matrix[i][j];`.
3. Trả về `transposedMatrix`

For example:

Input	Result
4 3 11 0 3 5 5 4 9 5 8 5 5 1	11 5 9 5 0 5 5 5 3 4 8 1

Answer: (penalty regime: 0 %)

```
int** transpose(int** matrix, int nRows, int nCols) {
  int** transposed = new int*[nCols];
  for (int i = 0; i < nCols; ++i) {
    transposed[i] = new int[nRows];
    for (int j = 0; j < nRows; ++j) {
      transposed[i][j] = matrix[j][i];
    }
  }
  return transposed;
}
```

	Input	Expected	Got	
✓	4 3 11 0 3 5 5 4 9 5 8 5 5 1	11 5 9 5 0 5 5 5 3 4 8 1	11 5 9 5 0 5 5 5 3 4 8 1	✓
✓	7 3 9 13 18 12 1 4 6 3 12 2 2 6 18 15 5 16 9 7 12 10 7	9 12 6 2 18 16 12 13 1 3 2 15 9 10 18 4 12 6 5 7 7	9 12 6 2 18 16 12 13 1 3 2 15 9 10 18 4 12 6 5 7 7	✓

	Input	Expected	Got	
✓	31 21 456 239 209 306 221 65 404 160 74 470 315 98 267 613 307 428 146 135 366 325 341 426 152 227 428 300 586 76 515 181 190 155 452 8 246 383 380 523 344 524 623 532 162 501 227 557 517 386 75 419 391 346 391 586 160 244 222 554 517 100 445 293 593 271 244 8 589 634 50 341 403 186 454 354 265 531 414 491 140 408 622 639 302 68 550 136 145 378 193 150 524 200 21 265 638 119 142 413 603 628 205 100 19 209 25 447 588 397 520 141 172 29 66 559 28 86 633 185 244 616 297 294 270 103 159 412 457 300 187 220 92 91 625 623 330 297 629 316 126 112 494 367 566 135 416 0 277 541 226 570 471 116 648 589 312 630 259 557 414 415 198 221 241 642 4 467 137 33 54 350 245 239 339 63 68 269 473 426 37 387 581 91 323 273 192 217 190 319 325 619 13 284 364 631 587 377 382 253 253 262 102 571 388 327 440 38 235 191 187 386 633 136 302 261 188 631 507 638 353 171 582 427 206 357 147 78 308 157 7 235 115 320 39 358 82 600 58 496 69 590 190 141 555 83 319 535 390 261 362 604 106 242 580 54 205 219 587 287 610 206 165 324 424 628 177 67 240 616 322 290 627 460 104 563 525 610 279 317 135 633 586 125 154 322 309 488 87 382 52 435 521 121 428 308 425 617 543 236 412 269 312 340 31 400 89 613 250 21 91 214 96 39 633 67 360 285 634 569 231 613 645 259 183 184 558 494 489 288 353 44 42 307 25 66 12 479 537 166 435 64 431 465 584 421 59 111 594 370 550 102 7 444 439 562 512 583 59 488 401 494 185 627 602 591 620 335 325 434 225 172 9 227 40 597 534 634 215 590	456 426 162 271 550 447 457 541 54 619 633 320 580 563 425 285 537 488 650 380 111 556 61 170 106 236 426 469 532 246 488 239 152 501 244 136 588 300 226 350 13 136 39 54 525 617 634 166 401 43 298 32 451 233 576 632 596 204 36 20 371 0 209 227 227 8 145 397 187 570 245 284 302 358 205 610 543 569 435 494 82 37 637 128 53 630 633 207 191 188 491 150 422 306 428 557 589 378 520 220 471 239 364 261 82 219 279 236 231 64 185 636 167 606 105 308 106 247 14 538 473 165 349 122 221 300 517 634 193 141 92 116 339 631 188 600 587 317 412 613 431 627 386 116 519 335 23 314 521 212 508 451 488 440 526 65 586 386 50 150 172 91 648 63 587 631 58 287 135 269 645 465 602 47 121 21 260 649 269 529 565 458 485 33 199 354 404 76 75 341 524 29 625 589 68 377 507 496 610 633 312 259 584 591 22 233 528 271 73 616 72 314 109 626 214 576 309 160 515 419 403 200 66 623 312 269 382 638 69 206 586 340 183 421 620 118 607 285 502 559 595 553 502 599 284 584 268 623 74 181 391 186 21 559 330 630 473 253 353 590 165 125 31 184 59 335 594 45 511 148 254 96 385 85 570 524 541 404 462 470 190 346 454 265 28 297 259 426 253 171 190 324 154 400 558 111 325 545 492 418 286 148 547 396 224 309 68 139 168 49 315 155 391 354 638 86 629 557 37 262 582 141 424 322 89 494 594 434 518 507 233 111 259 559 506 249 268 132 338 593 379 98 452 586 265 119 633 316 414 387 102 427 555 628 309 613 489 370 225 282 39 381 266 337 453 412 426 605 371 461 161 27 267 8 160 531 142 185 126 415 581 571 206 83 177 488 250 288 550 172 343 370 473 6 67 4 224 650 226 46 4 99 550 613 246 244 414 413 244 112 198 91 388 357 319 67 87 21 353 102 9 424 261 6 312 11 112 39 39 149 296 436 620 332 307 383 222 491 603 616 494 221 323 327 147 535 240 382 91 44 7 227 122 530 233 339 490 279 272 283 595 181 349 631 218 428 380 554 140 628 297 367 241 273 440 78 390 616 52 214 42 444 40 214 48 295 443 98 263 551 593 70 362 403 292 638 146 523 517 408 205 294 566 642 192 38 308 261 322 435 96 307 439 597 304 371 237 214 208 222 406 606 233 514 104 342 204 135 344 100 622 100 270 135 4 217 235 157 362 290 521 39 25 562 534 330 126 364 412 599 382 647 543 535 87 569 476 397 366 524 445 639 19 103 416 467 190 191 7	456 426 162 271 550 447 457 541 54 619 633 320 580 563 425 285 537 488 650 380 111 556 61 170 106 236 426 469 532 246 488 239 152 501 244 136 588 300 226 350 13 136 39 54 525 617 634 166 401 43 298 32 451 233 576 632 596 204 36 20 371 0 209 227 227 8 145 397 187 570 245 284 302 358 205 610 543 569 435 494 82 37 637 128 53 630 633 207 191 188 491 150 422 306 428 557 589 378 520 220 471 239 364 261 82 219 279 236 231 64 185 636 167 606 105 308 106 247 14 538 473 165 349 122 221 300 517 634 193 141 92 116 339 631 188 600 587 317 412 613 431 627 386 116 519 335 23 314 521 212 508 451 488 440 526 65 586 386 50 150 172 91 648 63 587 631 58 287 135 269 645 465 602 47 121 21 260 649 269 529 565 458 485 33 199 354 404 76 75 341 524 29 625 589 68 377 507 496 610 633 312 259 584 591 22 233 528 271 73 616 72 314 109 626 214 576 309 160 515 419 403 200 66 623 312 269 382 638 69 206 586 340 183 421 620 118 607 285 502 559 595 553 502 599 284 584 268 623 74 181 391 186 21 559 330 630 473 253 353 590 165 125 31 184 59 335 594 45 511 148 254 96 385 85 570 524 541 404 462 470 190 346 454 265 28 297 259 426 253 171 190 324 154 400 558 111 325 545 492 418 286 148 547 396 224 309 68 139 168 49 315 155 391 354 638 86 629 557 37 262 582 141 424 322 89 494 594 434 518 507 233 111 259 559 506 249 268 132 338 593 379 98 452 586 265 119 633 316 414 387 102 427 555 628 309 613 489 370 225 282 39 381 266 337 453 412 426 605 371 461 161 27 267 8 160 531 142 185 126 415 581 571 206 83 177 488 250 288 550 172 343 370 473 6 67 4 224 650 226 46 4 99 550 613 246 244 414 413 244 112 198 91 388 357 319 67 87 21 353 102 9 424 261 6 312 11 112 39 39 149 296 436 620 332 307 383 222 491 603 616 494 221 323 327 147 535 240 382 91 44 7 227 122	✓

	Input	Expected	Got	
	650 43 82 636 386 47 22 118 594 545 518 282 343 424 122 214 304 330 532 164 190 380 298 37 167 116 121 233 607 45 492 507 39 370 261 530 48 371 126 169 174 68 111 32 637 606 519 21 528 285 511 418 233 381 473 6 233 295 237 364 46 252 567 556 451 128 105 335 260 271 502 148 286 111 266 6 312 339 443 214 412 508 35 568 61 233 53 308 23 649 73 559 254 148 259 337 67 11 490 98 208 599 391 501 79 170 576 630 106 314 269 616 595 96 547 559 453 4 112 279 263 222 382 593 419 482 106 632 633 247 521 529 72 553 385 396 506 412 224 39 272 551 406 647 347 597 337 236 596 207 14 212 565 314 502 85 224 249 426 650 39 283 593 606 543 464 371 456 426 204 191 538 508 458 109 599 570 309 268 605 226 149 595 70 233 535 83 351 352 469 36 188 473 451 485 626 284 524 68 132 371 46 296 181 362 514 87 195 406 378 532 20 491 165 488 33 214 584 541 139 338 461 4 436 349 403 104 569 418 286 410 246 371 150 349 440 199 576 268 404 168 593 161 99 620 631 292 342 476 494 135 552 488 0 422 122 526 354 309 623 462 49 379 27 550 332 218 638 204 397 297 450 515	604 627 121 633 66 512 634 532 169 46 508 391 593 347 464 83 195 418 494 297 325 623 293 302 209 159 0 137 319 187 235 106 460 428 67 12 583 215 164 174 252 35 501 419 597 371 351 406 286 135 450 341 532 593 68 25 412 277 33 325 386 115 242 104 308 360 479 59 590 190 68 567 568 79 482 337 456 352 378 410 552 515	530 233 339 490 279 272 283 595 181 349 631 218 428 380 554 140 628 297 367 241 273 440 78 390 616 52 214 42 444 40 214 48 295 443 98 263 551 593 70 362 403 292 638 146 523 517 408 205 294 566 642 192 38 308 261 322 435 96 307 439 597 304 371 237 214 208 222 406 606 233 514 104 342 204 135 344 100 622 100 270 135 4 217 235 157 362 290 521 39 25 562 534 330 126 364 412 599 382 647 543 535 87 569 476 397 366 524 445 639 19 103 416 467 190 191 7 604 627 121 633 66 512 634 532 169 46 508 391 593 347 464 83 195 418 494 297 325 623 293 302 209 159 0 137 319 187 235 106 460 428 67 12 583 215 164 174 252 35 501 419 597 371 351 406 286 135 450 341 532 593 68 25 412 277 33 325 386 115 242 104 308 360 479 59 590 190 68 567 568 79 482 337 456 352 378 410 552 515	
✓	7 14 69 53 23 58 83 96 17 63 66 48 30 50 80 83 18 20 60 21 74 46 19 54 18 26 43 82 75 44 94 60 38 70 58 7 22 96 43 86 86 16 53 26 69 76 68 25 38 57 5 51 77 87 48 18 63 13 46 69 90 89 16 26 13 35 37 94 46 90 21 11 44 1 22 97 44 73 65 43 6 14 12 69 27 25 71 26 60 88 56 87 55 1 71 70 71 48 39 10	69 18 94 69 46 44 71 53 20 60 76 69 1 26 23 60 38 68 90 22 60 58 21 70 25 89 97 88 83 74 58 38 16 44 56 96 46 7 57 26 73 87 17 19 22 5 13 65 55 63 54 96 51 35 43 1 66 18 43 77 37 6 71 48 26 86 87 94 14 70 30 43 86 48 46 12 71 50 82 16 18 90 69 48 80 75 53 63 21 27 39 83 44 26 13 11 25 10	69 18 94 69 46 44 71 53 20 60 76 69 1 26 23 60 38 68 90 22 60 58 21 70 25 89 97 88 83 74 58 38 16 44 56 96 46 7 57 26 73 87 17 19 22 5 13 65 55 63 54 96 51 35 43 1 66 18 43 77 37 6 71 48 26 86 87 94 14 70 30 43 86 48 46 12 71 50 82 16 18 90 69 48 80 75 53 63 21 27 39 83 44 26 13 11 25 10	✓

	Input	Expected	Got	
✓	1 7 1 0 4 4 0 2 2	1 0 4 4 0 2 2	1 0 4 4 0 2 2	✓

Passed all tests! ✓

Đúng

Marks for this submission: 10,00/10,00.

Câu hỏi 9

Đúng

Đạt điểm 10,00 trên 10,00

Yêu cầu:

Viết hàm đảo ngược xâu dùng con trỏ `void reverse(char *s)`.

Input:

- Hàm nhận vào một xâu ký tự có kiểu dữ liệu là `char*` là một con trỏ đến một xâu ký tự (không chứa dấu cách) có độ dài không quá 50 ký tự.

Output:

- Hàm không có giá trị trả về

Gợi ý:

1. Đổi chỗ ký tự đầu tiên và ký tự cuối cùng. Lặp lại thao tác với ký tự thứ hai và ký tự trước cuối cùng, v.v.
2. Tạo hai con trỏ `char *left, *right` lần lượt trỏ tới ký tự đầu và cuối của xâu.
3. Bắt đầu vòng lặp while
4. Dùng hàm swap để đổi chỗ left và right.
5. `left++` và `right--` để di chuyển hai con trỏ.
6. Vòng lặp kết thúc khi `right <= left`

Lưu ý:

1. **KHÔNG** dùng thêm thư viện.
2. Các thư viện có sẵn gồm: `<iostream>` và `<cstring>` (chỉ dùng hàm lấy độ dài chuỗi).

For example:

Test	Input	Result

`char *s;`
`s = new char[50];`
`cin >> s;`
`reverse(s);`
`cout << s;`
	abc	cba

`char *s;`
`s = new char[50];`
`cin >> s;`
`reverse(s);`
`cout << s;`
	123456	654321

Answer: (penalty regime: 0 %)

Reset answer

```
1  void reverse(char *s) {
2    int lo = 0;
3    int hi = strlen(s) - 1;
4    while (lo < hi) {
5      s[lo] = s[lo] ^ s[hi];
6      s[hi] = s[lo] ^ s[hi];
7      s[lo] = s[lo] ^ s[hi];
8      lo++;
9      hi--;
10   }
11 }
12
```

	Test	Input	Expected	Got

✓	char *s;
	s = new char[50];
	cin >> s;
	reverse(s);
	cout << s;
		abc	cba	cba

✓	char *s;
	s = new char[50];
	cin >> s;
	reverse(s);
	cout << s;
		123456	654321	654321

✓	char *s;
	s = new char[50];
	cin >> s;
	reverse(s);
	cout << s;
		abcdef	fedcba	fedcba

✓	char *s;
	s = new char[50];
	cin >> s;
	reverse(s);
	cout << s;
		a	a	a

✓	char *s;
	s = new char[50];
	cin >> s;
	reverse(s);
	cout << s;
		12321	12321	12321

✓	char *s;
	s = new char[50];
	*(s) = '\0';
	reverse(s);
	cout << ":" << s;
			:	:

	Test	Input	Expected	Got

✓	`char *s;`
	`s = new char[50];`
	`cin >> s;`
	`reverse(s);`
	`cout << s;`
		ab	ba	ba

✓	`char *s;`
	`s = new char[50];`
	`for (int i=0; i<49; i++) {`
	`    *(s+i) = 'a';`
	`}`
	`*(s+49) = '\0';`
	`reverse(s);`
	`cout << ":" << s << ":";`
			:aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa:	:aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

Passed all tests! ✓

Đúng

Marks for this submission: 10,00/10,00.

Câu hỏi **10**

Đúng

Đạt điểm 10,00 trên 10,00

Yêu cầu:

Viết hàm lọc ký tự phía bên phải xâu dùng con trỏ `void rFilter(char *s)`.

Lọc các ký tự không nằm trong bảng chữ cái tiếng Anh (a-zA-Z) nằm ngoài cùng bên phải xâu, chuyển thành '_'.

Input:

- Hàm nhận vào một xâu ký tự có kiểu dữ liệu là `char*` là một con trỏ đến một xâu ký tự (không chứa dấu cách) có độ dài không quá 50 ký tự.

Output:

- Hàm không có giá trị trả về

Gợi ý:

1. Dùng con trỏ duyệt từ phải sang trái. Nếu ký tự không nằm trong bảng chữ cái thì thay thế.
2. Tạo hai con trỏ `char *right` trỏ tới ký tự cuối của xâu.
3. Bắt đầu vòng lặp while
4. Kiểm tra ký tự nằm trong bảng chữ cái. Ví dụ `'a' <= c && 'z' >= c && 'A' <= c && 'Z' >= c`.
5. Nếu không thoả mãn thì thay thế.
6. Vòng lặp kết thúc khi điều kiện trên không thoả mãn.
7. Có thể không cần dùng if và để điều kiện trực tiếp trong vòng while.

Lưu ý:

1. **KHÔNG** dùng thêm thư viện.
2. Các thư viện có sẵn gồm: `<iostream>` và `<cstring>` (chỉ dùng hàm lấy độ dài chuỗi).

For example:

Test	Input	Result

`char *s;`
`s = new char[50];`
`cin >> s;`
`rFilter(s);`
`cout << s;`
	`abc*(^$`	`abc____`

`char *s;`
`s = new char[50];`
`cin >> s;`
`rFilter(s);`
`cout << s;`
	`THCS4`	`THCS_`

Answer: (penalty regime: 0 %)

Reset answer

```
void rFilter(char *s) {
  for (int i = strlen(s) - 1; i >= 0; --i) {
    if (('a' <= s[i] && s[i] <= 'z') || ('A' <= s[i] && s[i] <= 'Z')) {
      break;
    }
    s[i] = '_';
  }
}
```

	Test	Input	Expected	Got

✓	char *s;
	s = new char[50];
	cin >> s;
	rFilter(s);
	cout << s;
		abc*(^$	abc____	abc____

✓	char *s;
	s = new char[50];
	cin >> s;
	rFilter(s);
	cout << s;
		THCS4	THCS_	THCS_

✓	char *s;
	s = new char[50];
	cin >> s;
	rFilter(s);
	cout << s;
		abcdef	abcdef	abcdef

✓	char *s;
	s = new char[50];
	cin >> s;
	rFilter(s);
	cout << s;
		12345	_____	_____

✓	char *s;
	s = new char[50];
	cin >> s;
	rFilter(s);
	cout << s;
		123CatCCC4	123CatCCC_	123CatCCC_

✓	char *s;
	s = new char[50];
	*(s) = '\0';
	rFilter(s);
	cout << ":" << s;
			:	:

	Test	Input	Expected	Got

✓	`char *s;`
	`s = new char[50];`
	`cin >> s;`
	`rFilter(s);`
	`cout << s;`
		`123CATcd4`	`123CATcd_`	`123CATcd_`

✓	`char *s;`
	`s = new char[50];`
	`for (int i=0; i<49; i++) {`
	`*(s+i) = 'a';`
	`}`
	`*(s+49) = '\0';`
	`rFilter(s);`
	`cout << ":" << s << ":";`
			`:aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa:`	`:aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa`

Passed all tests! ✓

Đúng

Marks for this submission: 10,00/10,00.

Trở lại Khoá học