

Status	Finished
Started	Tuesday, 10 December 2024, 1:49 PM
Completed	Saturday, 21 December 2024, 9:14 PM
Duration	11 days 7 hours
Marks	140.00/140.00
Grade	10.00 out of 10.00 (100%)

## Question 1

Correct

Mark 10.00 out of 10.00

**[RightAdding]**

Viết hàm `void pad_right(char *s, int n)` với nhiệm vụ điền thêm các kí tự '\_' vào cuối xâu `s` để `s` có độ dài bằng `n`. Nếu xâu `s` có độ dài lớn `n` thì không phải làm gì cả.

Ví dụ, hàm `pad_right` nhận vào xâu `s = "abc"` và `n = 5` thì khi kết thúc hàm `pad_right`, xâu `s` có giá trị là `"abc__"`

**Answer:** (penalty regime: 0 %)

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  #define ll long long
4  #define ull unsigned long long
5  #define el "\n"
6  #define se second
7  #define fi first
8  #define en end()
9  #define be begin()
10 #define sz size()
11 #define Faster ios_base::sync_with_stdio(0); cin.tie(0); cout.tie(0);
12
13 void pad_right(char *s, int n)
14 {
15     int len = strlen(s);
16     //for(int i = 0; i < len; i++) cout << s[i];
17     //for(int i = len; i < n; i++) cout << "_";
18     while(len < n)
19     {
20         s[len] = '_';
21         s[++len] = '\0';
22     }
23 }
24 
```

	Input	Expected	Got	
✓	abc 5	abc__	abc__	✓
✓	abccc 3	abccc	abccc	✓

Passed all tests! ✓

Correct

Marks for this submission: 10.00/10.00.

## Question 2

Correct

Mark 10.00 out of 10.00

## [LeftAdding]

Viết hàm `void pad_left(char *s, int n)` với nhiệm vụ điền thêm các kí tự '\_' vào đầu chuỗi `s` để `s` có độ dài bằng `n`. Nếu chuỗi `s` có độ dài lớn `n` thì không phải làm gì cả.

Ví dụ, hàm `pad_left` nhận vào chuỗi `s = "abc"` và `n = 5` thì khi kết thúc hàm `pad_left`, chuỗi `s` có giá trị là `"__abc"`

**Answer:** (penalty regime: 0 %)

```
1 #include<bits/stdc++.h>
2 void pad_left(char *s, int n)
3 {
4     int len = strlen(s);
5     for(int i = len; i < n; i++) cout << "_";
6 }
```

	Input	Expected	Got	
✓	abc 5	__abc	__abc	✓
✓	abccc 3	abccc	abccc	✓

Passed all tests! ✓

Correct

Marks for this submission: 10.00/10.00.

**Question 3**

Correct

Mark 10.00 out of 10.00

## [trim\_right]

Viết hàm `void trim_right(char *s)` với nhiệm vụ xóa các kí tự trắng đứng ở cuối chuỗi `s` nếu có.

Ví dụ, hàm `trim_right` nhận vào chuỗi `s = "abc "` thì sau khi kết thúc hàm `trim_right`, chuỗi `s` sẽ có giá trị là `"abc"`

**Answer:** (penalty regime: 0 %)

```
1 #include<bits/stdc++.h>
2 void trim_right(char *s)
3 {
4     int len = strlen(s);
5     for(int i = len - 1; i >= 0; i++)
6     {
7         if(s[i] == ' ') s[i] = '\0';
8         else break;
9     }
10 }
```

	Input	Expected	Got	
✓	abc	abc	abc	✓
✓	abc	abc	abc	✓

Passed all tests! ✓

Correct

Marks for this submission: 10.00/10.00.

## Question 4

Correct

Mark 10.00 out of 10.00

**[Trim\_Left]**

Viết hàm `void trim_left(char *s)` với nhiệm vụ xóa các kí tự trắng đứng ở đầu chuỗi `s` nếu có.

Ví dụ, hàm `trim_left` nhận vào chuỗi `s = " abc"` thì sau khi kết thúc hàm `trim_left`, chuỗi `s` sẽ có giá trị là `"abc"`

**Answer:** (penalty regime: 0 %)

```

1  #include<bits/stdc++.h>
2  void trim_left(char *s)
3  {
4      int size = strlen(s);
5      vector<char>vt;
6      for(int i = 0; i < size; i++)
7      {
8          if(s[i] != ' ') vt.push_back(s[i]);
9      }
10     int len = vt.size();
11     for(int i = 0; i < len; i++)
12     {
13         s[i] = vt[i];
14     }
15     for(int i = len; i < size; i++)
16     {
17         s[i] = '\0';
18     }
19 }
```

	Input	Expected	Got	
✓	abc	abc	abc	✓
✓	abc	abc	abc	✓

Passed all tests! ✓

Correct

Marks for this submission: 10.00/10.00.

## Question 5

Correct

Mark 10.00 out of 10.00

**[Reverse]**

Viết một hàm `void reverse(char *s)` nhận vào một chuỗi và đảo ngược thứ tự của chuỗi `s` đó.

Ví dụ, hàm `reverse` nhận vào chuỗi `s = "abc"`. Khi kết thúc hàm `reverse` chuỗi `s` sẽ có giá trị là `"cba"`.

**Answer:** (penalty regime: 0 %)

```
1 #include<bits/stdc++.h>
2 using namespace std;
3 void reverse(char *s)
4 {
5     int i = 0, j = strlen(s) - 1;
6     while(i < j)
7     {
8         swap(s[i++], s[j--]);
9     }
10 }
11 }
```

	Input	Expected	Got	
✓	abcdef	fedcba	fedcba	✓
✓	zxcvb	bvcxz	bvcxz	✓

Passed all tests! ✓

Correct

Marks for this submission: 10.00/10.00.

Question 6

Correct

Mark 10.00 out of 10.00

[GetSum]

Viết hàm `int getSum(int *a, int n)` nhận vào mảng số nguyên `a` có `n` phần tử và tính tổng giá trị của các phần tử trong mảng `a`.

Answer: (penalty regime: 0 %)

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  #define el "\n"
4  #define ll long long
5  #define ull unsigned long long
6  #define se second
7  #define fi first
8  #define be begin()
9  #define en end()
10 #define Faster cin.tie(0); cout.tie(0); ios_base::sync_with_stdio(0);
11
12 int getSum(int *a, int n)
13 {
14     ll sum = 0;
15     for(int i = 0; i < n; i++)
16     {
17         cin >> a[i];
18         sum += a[i];
19     }
20     return sum;
21 }
```

	Input	Expected	Got	
✓	3 1 2 3	6	6	✓
✓	10 77 42 60 77 54 9 84 43 54 94	594	594	✓
✓	30 83 77 26 39 81 60 94 67 46 15 12 8 93 96 71 15 84 11 29 86 73 64 63 97 91 94 46 34 48 1	1704	1704	✓
✓	50 84 98 60 37 53 16 44 25 18 65 26 74 70 44 0 8 42 69 13 55 74 21 71 41 31 36 51 93 50 46 15 75 12 12 1 21 43 85 42 42 88 77 5 21 21 4 96 21 29 97	2222	2222	✓

	Input	Expected	Got	
✓	100 56 87 56 42 66 68 14 61 25 48 73 43 5 15 7 30 13 8 53 9 71 54 17 22 16 85 19 10 87 66 0 32 15 76 46 52 33 92 8 29 43 80 65 8 81 9 10 8 13 18 57 83 78 7 67 38 45 23 90 60 80 26 8 11 86 38 6 32 34 96 76 16 63 2 8 70 92 58 89 70 25 9 38 79 22 64 7 66 23 19 99 91 52 29 59 51 9 8 54 14	4271	4271	✓

Passed all tests! ✓

Correct

Marks for this submission: 10.00/10.00.



## Question 7

Correct

Mark 10.00 out of 10.00

**[Delete\_Char]**

Viết hàm `void delete_char(char *S, char c)` nhận vào chuỗi  $S$  và ký tự  $c$ , sau đó hàm này sẽ xóa hết các ký tự có giá trị bằng  $c$  trong chuỗi  $S$ .

Ví dụ, hàm `delete_char` nhận vào chuỗi  $S = "abcdac"$  và ký tự  $c = 'a'$  thì sau khi kết thúc hàm,  $S$  sẽ nhận giá trị là `"bcd"`.

For example:

Input	Result
abcccaa a	bccc

Answer: (penalty regime: 0 %)

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  #define el "\n"
4  #define ll long long
5  #define ull unsigned long long
6  #define se second
7  #define fi first
8  #define be begin()
9  #define en end()
10 #define Faster cin.tie(0); cout.tie(0); ios_base::sync_with_stdio(0);
11
12 void delete_char(char *S, char c)
13 {
14     int Size = strlen(S);
15     string s = "";
16     for(int i = 0; i < Size; i++)
17     {
18         if(S[i] != c) s += S[i];
19     }
20     Size = s.size();
21     for(int i = 0; i < Size; i++) S[i] = s[i];
22     S[Size] = '\0';
23     //for(int i = 0; i < s.size(); i++) cout << S[i];
24
25 }
```

	Input	Expected	Got	
✓	abcccaa a	bccc	bccc	✓

Passed all tests! ✓

Correct

Marks for this submission: 10.00/10.00.



## Question 8

Correct

Mark 10.00 out of 10.00

**[Merge]**

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.

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*.

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.

Lưu ý: các mảng đầu vào có thể [sắp xếp tăng dần](#) hoặc giảm dần.

**For example:**

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

**Answer:** (penalty regime: 0 %)

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  #define el "\n"
4  #define ll long long
5  #define ull unsigned long long
6  #define se second
7  #define fi first
8  #define be begin()
9  #define en end()
10 #define Faster cin.tie(0); cout.tie(0); ios_base::sync_with_stdio(0);
11
12 int* merge(int* firstArr, int lenArr1, int* secondArr, int lenArr2)
13 {
14     int* ptr = new int [lenArr1 + lenArr2];
15     vector<int> vt;
16     int i = 0, j = 0;
17     while(i < lenArr1 && j < lenArr2)
18     {
19         if(firstArr[i] <= secondArr[j])
20         {
21             vt.push_back(firstArr[i++]);
22         }
23         else
24         {
25             vt.push_back(secondArr[j++]);
26         }
27     }
28     while(i < lenArr1)
29     {
30         vt.push_back(firstArr[i++]);
31     }
32     while(j < lenArr2)
33     {
34         vt.push_back(secondArr[j++]);
35     }
36     if(firstArr[0] < firstArr[lenArr1 - 1]) sort(vt.be, vt.en);
37     else sort (vt.be, vt.en, greater<int>());
38     for(int t = 0; t < lenArr1 + lenArr2; t++)
39     {
40         ptr[t] = vt[t];

```

```
41     }
42     return ptr;
43 }
44
45
46
47
```

	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	✓

	Input	Expected	Got	
✓	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! ✓

Correct

Marks for this submission: 10.00/10.00.

## Question 9

Correct

Mark 10.00 out of 10.00

**[GetDataFromPointer]**

Viết chương trình nhập vào từ bàn phím một số nguyên bằng con trỏ và in lại giá trị số nguyên đó ra màn hình.

**For example:**

Input	Result
6	6

**Answer:** (penalty regime: 0 %)

Reset answer

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  #define el "\n"
4  #define ll long long
5  #define ull unsigned long long
6  #define se second
7  #define fi first
8  #define be begin()
9  #define en end()
10 #define Faster cin.tie(0); cout.tie(0); ios_base::sync_with_stdio(0);
11 void Run()
12 {
13     int x;
14     int *ptr = &x;
15     cin >> *ptr;
16     cout << *ptr;
17 }
18 int main()
19 {
20     Faster;
21     Run();
22     return 0;
23 }
```

Passed all tests! ✓

Correct

Marks for this submission: 10.00/10.00.

## Question 10

Correct

Mark 10.00 out of 10.00

**[getPointerToTen]**

Viết hàm `int* getPointerToTen()` . Hàm này khai báo một con trỏ kiểu nguyên, cấp phát bộ nhớ của một số nguyên cho con trỏ đó và gán giá trị 10 cho vùng bộ nhớ đó. Hàm trả về con trỏ đã khai báo.

**Answer:** (penalty regime: 0 %)

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  #define el "\n"
4  #define ll long long
5  #define ull unsigned long long
6  #define se second
7  #define fi first
8  #define be begin()
9  #define en end()
10 #define Faster cin.tie(0); cout.tie(0); ios_base::sync_with_stdio(0);
11 int* getPointerToTen()
12 {
13     int *ptr = new int;
14     *ptr = 10;
15     return ptr;
16 }
```

Passed all tests! ✓

Correct

Marks for this submission: 10.00/10.00.

## Question 11

Correct

Mark 10.00 out of 10.00

**[PrintImage]**

Một ảnh xám có thể biểu diễn bởi một ma trận hai chiều, trong đó mỗi phần tử của mảng biểu diễn một điểm ảnh (có giá trị từ 0 - 255). Giả sử ta muốn in một ảnh xám ra màn hình dòng lệnh, ta có thể in giá trị của từng điểm ảnh. Viết hàm `void printImage(int** img, int height, int width)` nhận tham số truyền vào là một con trỏ trỏ đến [mảng hai chiều](#) biểu diễn ảnh, chiều cao và chiều rộng của ảnh. Hàm in ra màn hình giá trị của từng điểm ảnh theo từng hàng, trên một hàng các giá trị luôn chiếm 4 khoảng trống.

Gợi ý: sử dụng `cout << setw(4) << number;` (C++) hoặc `printf("%4d", number);` (C/C++).

**Answer:** (penalty regime: 0 %)

```
1 void printImage(int** img, int height, int width)
2 {
3     for(int i = 0; i < height; i++)
4     {
5         for(int j = 0; j < width; j++)
6         {
7             cout << setw(4) << img[i][j];
8         }
9         cout << endl;
10    }
11 }
```



	Input	Expected	Got	
✓	20 20 255 0 255 255 0 0 0 0 0 0 0 0 255 122 122 255 0 0 0 0 0 0 0 0 255 255 0 0 0 0 255 255 255 255 122 122 255 255 255 255 0 0 0 0 255 255 0 0 0 122 255 255 255 122 122 122 122 255 255 255 122 0 0 0 255 255 0 0 255 122 122 122 122 122 122 122 122 122 122 122 122 255 0 0 255 255 0 0 255 181 122 122 122 181 181 181 181 122 122 122 181 255 0 0 255 255 0 0 255 255 122 122 122 255 255 255 255 122 122 122 255 255 0 0 255 255 0 255 255 255 122 246 255 255 255 255 255 255 181 122 255 255 255 0 255 255 0 255 255 255 122 246 255 255 255 255 255 255 181 122 255 255 255 0 255 255 0 255 255 122 122 246 255 255 255 255 255 122 122 122 255 122 0 255 255 0 122 122 0 0 0 0 0 0 0 0 0 0 0 0 0 122 122 0 255 255 0 113 113 0 17 17 17 0 17 17 0 17 17 17 0 113 113 0 255 255 0 0 0 0 255 255 255 0 255 255 0 255 255 255 0 0 0 0 255 255 0 0 0 255 255 255 255 0 255 255 0 255 255 255 255 0 0 0 255 255 0 0 0 255 255 255 255 255 0 0 0 255 255 255 255	255 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 255 255 0 0 0 0 0 0 0 255 122 122 255 0 0 0 0 0 0 0 255 255 0 0 0 0 255 255 255 255 122 122 255 255 255 255 0 0 0 0 255 255 0 0 0 122 255 255 255 122 122 122 122 255 255 255 122 0 0 0 255 255 0 0 255 122 122 122 122 122 122 122 122 122 122 122 255 0 0 255 255 0 0 255 181 122 122 122 181 181 181 181 122 122 122 181 255 0 0 255 255 0 0 255 255 122 122 122 255 255 255 255 122 122 122 255 255 0 0 255 255 0 255 255 255 122 246 255 255 255 255 255 255 181 122 255 255 0 255 255 0 255 255 255 122 246 255 255 255 255 255 255 181 122 255 255 0 255 255 0 255 255 122 122 246 255 255 255 255 255 255 181 122 255 255 0 255 255 0 122 122 122 122 122 255 255 255 255 122 122 122 122 122 0 255 255 0 122 122 0 0 0 0 0 0 0 0 0 0 0 0 122 122 0 255 255 0 113 113 0 17 17 17 0 17 17 0 17 17 17 0 113 113 0 255 255 0 0 0 0 255 255 255 0 255 255 0 255 255 255 0 0 0 0 255 255 0 0 0 255 255 255 255 0 255 255 0 255 255 255 255 0 0 255 255 0 0 0 255 255 255 255 0 255 255 0 0 0 255 255 255 255 255 255 255 255 255 255 255 0 0 255 0 0 0 255 255 255 255 255 255 255 255 255 255 255 0 0	255 0 255 255 0 0 0 0 0 0 0 0 255 122 122 255 0 0 0 0 0 0 0 0 255 255 0 0 0 0 255 255 255 255 255 122 122 255 255 255 255 0 0 0 0 255 255 0 0 255 122 122 122 122 122 122 122 122 122 122 122 122 255 0 0 255 255 0 0 255 181 122 122 122 181 181 181 181 122 122 122 181 255 0 0 255 255 0 0 255 255 122 122 122 255 255 255 255 122 122 122 255 255 0 255 255 0 255 255 255 122 246 255 255 255 255 255 181 122 255 255 0 255 255 0 255 255 122 122 246 255 255 255 255 255 181 122 255 255 255 0 255 255 0 255 255 255 122 246 255 255 255 255 255 181 122 255 255 255 0 255 255 0 122 122 122 122 122 122 255 255 255 255 122 122 122 122 122 122 0 255 255 0 122 122 0 0 0 0 0 0 0 0 0 0 0 0 122 122 0 255 255 0 113 113 0 17 17 17 0 17 17 0 17 17 17 0 113 113 0 255 255 0 0 0 0 255 255 255 0 255 255 0 255 255 255 0 0 0 0 255 255 0 0 0 255 255 255 255 0 255 255 0 255 255 255 255 0 0 0 255 255 0 0 0 255 255 255 255 255 255 255 255 255 255 255 0 0 255 0 0 0 255 255 255 255 255 255 255 255 255 255 255 255	✓

	Input	Expected	Got	
	255 255 255 255 255 255 255 255 0 0 0 255 255 0 0 0 0 255 255 255 255 255 255 255 255 255 255 0 0 0 0 255 255 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255	0 255 255 0 0 0 0 255 255 255 255 255 255 255 255 255 255 0 0 0 0 255 255 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255	0 0 0 255 255 0 0 0 0 255 255 255 255 255 255 255 255 255 255 0 0 0 0 255 255 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 255	
✓	16 16 255 0 0 255 255 0 0 255 255 255 255 255 255 255 255 255 0 29 29 0 0 29 29 0 255 255 255 255 255 255 255 255 0 29 29 0 0 29 29 0 255 255 255 255 255 255 0 0 29 255 29 29 29 29 29 29 0 0 255 255 255 255 0 0 29 29 29 29 29 29 29 29 0 0 255 255 255 255 0 0 16 29 29 29 29 29 29 16 0 0 255 255 255 255 0 0 16 29 29 29 29 29 29 16 0 0 255 255 255 255 255 255 0 16 29 29 29 29 16 0 255 255 255 255 255 255 255 255 255 0 16 29 29 16 0 255 255 255 255 255 255 255 255 255 255 255 0 16 16 0 255 255 255 255 255 255 255 255 255 255 255 255 255 0 0 255 255 255 255 255 255 255 255 255 255 255 255 255 255 0 0 255	255 0 0 255 255 0 0 255 255 255 255 255 255 255 255 255 0 29 29 0 0 29 29 0 255 255 255 255 255 255 255 255 0 29 29 0 0 29 29 0 255 255 255 255 255 255 0 0 29 255 29 29 29 29 29 29 0 0 255 255 255 255 0 0 29 29 29 29 29 29 29 29 0 0 255 255 255 255 0 0 16 29 29 29 29 29 29 16 0 0 255 255 255 255 0 0 16 29 29 29 29 255 255 255 255 0 16 29 29 29 29 16 0 255 255 255 255 255 255 255 255 255 255 0 16 16 0 255 255 255 255 255 255 255 255 255 255 255 255 255 0 0 255 255 255 255 255 255 255 255 255 255 255 255 255 255 0 0 255	255 0 0 255 255 255 255 255 0 29 29 0 255 255 255 255 0 29 29 0 0 29 29 0 255 255 255 255 0 29 29 0 255 255 255 255 255 255 0 0 29 255 29 29 29 29 29 29 0 0 255 255 255 255 0 0 29 29 29 29 29 29 29 29 0 0 255 255 255 255 255 255 0 16 29 29 255 255 0 0 16 29 29 29 29 29 29 16 0 0 255 255 255 255 255 255 0 16 29 29 29 29 16 0 255 255 255 255 255 255 255 255 255 0 16 29 29 16 0 255 255 255 255 16 16 0 255 255 255 255 255 255 255 255 255 255 255 255 255 0 255 255 255 255 255 255 255 0 255 255 255 255 255 255 255 0 255	✓

Passed all tests! ✓

Correct

Marks for this submission: 10.00/10.00.

## Question 12

Correct

Mark 10.00 out of 10.00

**[SelfDividingNumbers]**

Số tự phân chia là số chia hết cho mỗi chữ số của nó.

Ví dụ: 128 là số tự phân chia vì  $128 \% 1 == 0$ ,  $128 \% 2 == 0$ , và  $128 \% 8 == 0$ .

Ngoài ra, số tự phân chia không được phép chứa số 0.

Viết hàm `int* selfDividingNumbers(int left, int right, int* returnSize)` nhận đầu vào là một ngưỡng chặn dưới *left* và một ngưỡng chặn trên *right*. Hàm thực hiện tính toán và trả về danh sách các số tự phân chia nằm trong đoạn giới hạn bởi *left* và *right*.

Giá trị biến *returnSize* đại diện cho số các số tự phân chia của mảng trả về và giá trị của biến có thể thay đổi trong hàm.

**Ghi chú:** Tránh phép chia 0.

**For example:**

Input	Result
37 58	44 48 55

**Answer:** (penalty regime: 0 %)

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  #define el "\n"
4  #define ll long long
5  #define ull unsigned long long
6  #define se second
7  #define fi first
8  #define be begin()
9  #define en end()
10 #define Faster cin.tie(0); cout.tie(0); ios_base::sync_with_stdio(0);
11 bool check(int n)
12 {
13     string s = to_string(n);
14     int sIze = s.size();
15     for(int i = 0; i < sIze; i++)
16     {
17         if(s[i] == '0') return false;
18         if(n % (s[i] - '0')) return false;
19     }
20     return true;
21 }
22 int* selfDividingNumbers(int left, int right, int* returnSize)
23 {
24     vector<int> vt;
25     for(int i = left; i <= right; i++)
26     {
27         if(check(i)) vt.push_back(i);
28     }
29     int sIze = vt.size();
30     int *a = new int[sIze];
31     for(int i = 0; i < sIze; i++)
32     {
33         a[i] = vt[i];
34     }
35     *returnSize = sIze;
36     return a;
37 }
```

	Input	Expected	Got	
✓	29 93	33 36 44 48 55 66 77 88	33 36 44 48 55 66 77 88	✓
✓	37 58	44 48 55	44 48 55	✓
✓	47 520	48 55 66 77 88 99 111 112 115 122 124 126 128 132 135 144 155 162 168 175 184 212 216 222 224 244 248 264 288 312 315 324 333 336 366 384 396 412 424 432 444 448 488 515	48 55 66 77 88 99 111 112 115 122 124 126 128 132 135 144 155 162 168 175 184 212 216 222 224 244 248 264 288 312 315 324 333 336 366 384 396 412 424 432 444 448 488 515	✓
✓	5 283	5 6 7 8 9 11 12 15 22 24 33 36 44 48 55 66 77 88 99 111 112 115 122 124 126 128 132 135 144 155 162 168 175 184 212 216 222 224 244 248 264	5 6 7 8 9 11 12 15 22 24 33 36 44 48 55 66 77 88 99 111 112 115 122 124 126 128 132 135 144 155 162 168 175 184 212 216 222 224 244 248 264	✓
✓	78 86			✓

Passed all tests! ✓

Correct

Marks for this submission: 10.00/10.00.

## Question 13

Correct

Mark 10.00 out of 10.00

**[SquareWithPointer]**

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.

**Answer:** (penalty regime: 0 %)

```
1 double* getSquare (double number)
2 {
3     double *ptr = new double;
4     *ptr = number * number;
5     return ptr;
6 }
```

	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! ✓

Correct

Marks for this submission: 10.00/10.00.

## Question 14

Correct

Mark 10.00 out of 10.00

**[Transpose Matrix]**

**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 m, int n)` nhận đầu vào là một ma trận  $A$  (*matrix*) bất kỳ có kích cỡ  $m \times n$  chứa các giá trị nguyên.

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

**For example:**

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

**Answer:** (penalty regime: 0 %)

```
1 | int** transpose (int** matrix, int m, int n)
2 | {
3 |     int **mAtrix = new int* [n];
4 |     for(int i = 0; i < n; i++) mAtrix[i] = new int[m];
5 |     for(int i = 0 ; i < m; i++)
6 |     {
7 |         for(int j =0; j < n; j++) mAtrix[j][i] = matrix[i][j];
8 |     }
9 |     return mAtrix;
10| }
```

	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	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	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	✓

	Input	Expected	Got	
	421 59 111 594 370 550 102 7	208 222 406 606 233 514 104 342 204	613 246 244 414 413 244 112 198 91 388	
	444 439 562 512 583 59	135 344 100 622 100 270 135 4 217 235 157	357 319 67 87 21 353 102 9 424 261 6	
	488 401 494 185 627 602 591	362 290 521 39 25 562 534 330 126 364 412	312 11 112 39 39 149 296 436 620 332	
	620 335 325 434 225 172 9	599 382 647 543 535 87 569 476 397	307 383 222 491 603 616 494 221 323	
	227 40 597 534 634 215 590	366 524 445 639 19 103 416 467 190 191 7	327 147 535 240 382 91 44 7 227 122	
	650 43 82 636 386 47 22 118	604 627 121 633 66 512 634 532 169 46 508	530 233 339 490 279 272 283 595 181	
	594 545 518 282 343 424 122	391 593 347 464 83 195 418 494 297	349 631 218	
	214 304 330 532 164 190	325 623 293 302 209 159 0 137 319 187 235	428 380 554 140 628 297 367 241 273	
	380 298 37 167 116 121 233	106 460 428 67 12 583 215 164 174 252 35	440 78 390 616 52 214 42 444 40 214 48	
	607 45 492 507 39 370 261	501 419 597 371 351 406 286 135 450	295 443 98 263 551 593 70 362 403 292	
	530 48 371 126 169 174 68	341 532 593 68 25 412 277 33 325 386 115	638	
	111 32 637 606 519 21 528	242 104 308 360 479 59 590 190 68 567 568	146 523 517 408 205 294 566 642 192 38	
	285 511 418 233 381 473 6	79 482 337 456 352 378 410 552 515	308 261 322 435 96 307 439 597 304 371	
	233 295 237 364 46 252 567		237 214 208 222 406 606 233 514 104	
	556 451 128 105 335 260 271		342 204	
	502 148 286 111 266 6 312		135 344 100 622 100 270 135 4 217 235	
	339 443 214 412 508 35 568		157 362 290 521 39 25 562 534 330 126	
	61 233 53 308 23 649 73 559		364 412 599 382 647 543 535 87 569 476	
	254 148 259 337 67 11 490 98		397	
	208 599 391 501 79		366 524 445 639 19 103 416 467 190 191	
	170 576 630 106 314 269 616		7 604 627 121 633 66 512 634 532 169	
	595 96 547 559 453 4 112 279		46 508 391 593 347 464 83 195 418 494	
	263 222 382 593 419 482		297	
	106 632 633 247 521 529 72		325 623 293 302 209 159 0 137 319 187	
	553 385 396 506 412 224 39		235 106 460 428 67 12 583 215 164 174	
	272 551 406 647 347 597 337		252 35 501 419 597 371 351 406 286 135	
	236 596 207 14 212 565 314		450	
	502 85 224 249 426 650 39		341 532 593 68 25 412 277 33 325 386	
	283 593 606 543 464 371 456		115 242 104 308 360 479 59 590 190 68	
	426 204 191 538 508 458 109		567 568 79 482 337 456 352 378 410 552	
	599 570 309 268 605 226 149		515	
	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			

	Input	Expected	Got	
✓	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	✓
✓	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! ✓

Correct

Marks for this submission: 10.00/10.00.

[Back to Course](#)