Trạng thái	Đã xong
Bắt đầu vào lúc	Thứ Ba, 1 tháng 4 2025, 11:01 AM
Kết thúc lúc	Thứ Ba, 1 tháng 4 2025, 11:40 AM
Thời gian thực hiện	39 phút 23 giây
Điểm	60,00/60,00
Điểm	**10,00** trên 10,00 (**100**%)

Câu hỏi 1

Đúng

Đạt điểm 10,00 trên 10,00

Viết hàm `int** keepEven (int** matrix, int nRows, int nCols)` kiểm tra ma trận hai chiều.

Hàm nhận đầu vào là ma trận $matrix$ có kích thước $nRows \times nCols$. Hàm trả về một ma trận mới sao cho tất cả các giá trị là số lẻ trong ma trận ban đầu được gán giá trị 0 và giữ nguyên các số chẵn.

Answer: (penalty regime: 0 %)

```
int **keepEven(int **matrix, int nRows, int nCols){

    for(int i=0; i<nRows; i++){
        for(int j=0; j<nCols; j++){
            if(matrix[i][j]&1){
                matrix[i][j]=0;
            }
        }
    }
    return matrix;
}
```

	Input	Expected	Got	
✓	6 5 9 11 29 26 21 26 8 1 11 27 5 25 28 11 6 4 3 4 19 3 8 12 26 11 14 12 8 19 12 9	0 0 0 26 0 26 8 0 0 0 0 0 28 0 6 4 0 4 0 0 8 12 26 0 14 12 8 0 12 0	0 0 0 26 0 26 8 0 0 0 0 0 28 0 6 4 0 4 0 0 8 12 26 0 14 12 8 0 12 0	✓

	Input	Expected	Got	
✓	16 3 22 14 19 25 5 46 43 20 21 27 37 42 42 3 15 40 17 17 19 10 6 28 41 1 13 19 46 21 47 45 44 32 11 40 0 18 28 8 41 6 13 22 0 15 37 11 0 19	22 14 0 0 0 46 0 20 0 0 0 42 42 0 0 40 0 0 0 10 6 28 0 0 0 0 46 0 0 0 44 32 0 40 0 18 28 8 0 6 0 22 0 0 0 0 0 0	22 14 0 0 0 46 0 20 0 0 0 42 42 0 0 40 0 0 0 10 6 28 0 0 0 0 46 0 0 0 44 32 0 40 0 18 28 8 0 6 0 22 0 0 0 0 0 0	✓
✓	19 9 108 163 69 53 38 64 74 75 136 54 165 94 138 74 40 10 30 79 8 41 21 43 20 169 8 63 83 137 60 66 76 149 68 127 86 50 38 150 34 96 159 158 42 64 121 44 19 111 139 11 57 65 131 107 68 111 9 105 99 77 150 71 99 25 0 86 75 26 82 124 103 89 58 96 99 164 94 37 17 129 27 100 54 43 153 21 112 31 48 159 56 110 80 6 23 170 45 88 10 2 95 134 74 53 104 21 13 162 24 144 37 162 23 72 134 81 76 108 146 111 159 161 118 116 51 163 13 78 154 106 83 146 14 93 55 65 102 17 41 167 98 58 101 158 73 90 146 121 91 78 35 166 67 98 74 168 54 4 96 42 10 98 114 16 14 28 93 67 61 75 99	108 0 0 0 38 64 74 0 136 54 0 94 138 74 40 10 30 0 8 0 0 0 20 0 8 0 0 0 60 66 76 0 68 0 86 50 38 150 34 96 0 158 42 64 0 44 0 0 0 0 0 0 0 0 68 0 0 0 0 0 150 0 0 0 0 86 0 26 82 124 0 0 58 96 0 164 94 0 0 0 0 100 54 0 0 0 112 0 48 0 56 110 80 6 0 170 0 88 10 2 0 134 74 0 104 0 0 162 24 144 0 162 0 72 134 0 76 108 146 0 0 0 118 116 0 0 0 78 154 106 0 146 14 0 0 0 102 0 0 0 98 58 0 158 0 90 146 0 0 78 0 166 0 98 74 168 54 4 96 42 10 98 114 16 14 28 0 0 0 0 0	108 0 0 0 38 64 74 0 136 54 0 94 138 74 40 10 30 0 8 0 0 0 20 0 8 0 0 0 60 66 76 0 68 0 86 50 38 150 34 96 0 158 42 64 0 44 0 0 0 0 0 0 0 0 68 0 0 0 0 0 150 0 0 0 0 86 0 26 82 124 0 0 58 96 0 164 94 0 0 0 0 100 54 0 0 0 112 0 48 0 56 110 80 6 0 170 0 88 10 2 0 134 74 0 104 0 0 162 24 144 0 162 0 72 134 0 76 108 146 0 0 0 118 116 0 0 0 78 154 106 0 146 14 0 0 0 102 0 0 0 98 58 0 158 0 90 146 0 0 78 0 166 0 98 74 168 54 4 96 42 10 98 114 16 14 28 0 0 0 0 0	✓

Passed all tests! ✓

Đúng

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

Câu hỏi 2

Đúng

Đạt điểm 10,00 trên 10,00

Cho một dãy gồm n số nguyên và một ngưỡng nguyên (threshold). Viết hàm kiểm tra xem các số trong dãy cao hơn hay thấp hơn ngưỡng cho trước.

Hàm `bool* isHigher (int* arr, int num, int thres)` nhận đầu vào là mảng arr có n số nguyên và một ngưỡng $thres$.

Hàm kiểm tra và trả về một mảng số kiểu $bool$ với phần tử thứ i là $true$ nếu số nguyên thứ i trong mảng arr lớn hơn hoặc bằng ngưỡng $thres$, và bằng $false$ trong trường hợp ngược lại.

For example:

Input	Result
6 67 13 99 14 41 20 15	1 0 1 0 1 1

Answer: (penalty regime: 0 %)

```
bool *isHigher(int *arr, int num, int thres){
    bool *res=new bool [num];
    for(int i=0; i<num; i++){
        if(arr[i]>=thres){
            res[i]=true;
        }
        else{
            res[i]=false;
        }
    }
    return res;
}
```

	Input	Expected	Got	
✓	6 67 13 99 14 41 20 15	1 0 1 0 1 1	1 0 1 0 1 1	✓

	Input	Expected	Got	
✓	15 27 52 81 82 21 31 63 72 9 63 84 73 10 39 12 17	1 1 1 1 1 1 1 1 0 1 1 1 0 1 0	1 1 1 1 1 1 1 1 0 1 1 1 0 1 0	✓
✓	11 99 57 96 13 40 6 98 95 28 53 28 17	1 1 1 0 1 0 1 1 1 1 1	1 1 1 0 1 0 1 1 1 1 1	✓

Passed all tests! ✓

Đúng

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

Câu hỏi 3

Đúng

Đạt điểm 10,00 trên 10,00

Giả sử bạn được thuê xây dựng một hệ thống xử lý ảnh.

Trong đó có một bước lọc ảnh. Tại đây, bạn phải đọc ảnh vào và kiểm tra giá trị tại từng điểm ảnh (**pixel**) có lớn hơn hoặc bằng một ngưỡng **(threshold)** cố định không.

Nếu có, giá trị tại pixel đó giữ nguyên, nếu không, giá trị tại đó được gán giá trị bằng 0.

Viết hàm `int** getImage (int nRows, int nCols)` đọc vào một ảnh có kích thước $nRows \times nCols$. Hàm trả về một con trỏ trỏ tới con trỏ lưu trữ ảnh có kiểu **int****.

Viết hàm `void fillImage (int** image, int nRows, int nCols, int threshold)` thực hiện việc lọc ảnh với ngưỡng cho trước $threshold$. Hàm lọc và thay đổi giá trị của ảnh lưu bằng con trỏ $image$.

Cuối cùng, viết hàm `void print (int** image, int nRows, int nCols)` để in ra ảnh sau khi đã lọc.

Lưu ý: Các điểm ảnh là số nguyên có giá trị nằm trong khoảng từ 0 đến 255.

For example:

Input	Result
5 2 126 82 26 132 81 153 106 185 207 85 95	126 0 0 132 0 153 106 185 207 0

Answer: (penalty regime: 0 %)

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

void fillImage(int **image, int nRows, int nCols, int threshold){
    for(int i=0; i<nRows; i++){
        for(int j=0; j<nCols; j++){
            if(image[i][j]<threshold){
                image[i][j]=0;
            }
        }
    }
}

void print(int **image, int nRows, int nCols){
    for(int i=0; i<nRows; i++){
        for(int j=0; j<nCols; j++){
            cout<<image[i][j]<<" ";
        }
        cout<<endl;
    }
}
```

	Input	Expected	Got	
✓	5 2 126 82 26 132 81 153 106 185 207 85 95	126 0 0 132 0 153 106 185 207 0	126 0 0 132 0 153 106 185 207 0	✓
✓	16 3 217 64 12 91 107 46 70 204 109 128 222 184 194 216 127 112 49 134 214 69 101 166 93 38 240 190 151 152 59 225 72 214 48 35 0 139 31 132 253 137 28 218 195 226 1 93 172 25 17	217 64 0 91 107 46 70 204 109 128 222 184 194 216 127 112 49 134 214 69 101 166 93 38 240 190 151 152 59 225 72 214 48 35 0 139 31 132 253 137 28 218 195 226 0 93 172 25	217 64 0 91 107 46 70 204 109 128 222 184 194 216 127 112 49 134 214 69 101 166 93 38 240 190 151 152 59 225 72 214 48 35 0 139 31 132 253 137 28 218 195 226 0 93 172 25	✓

	Input	Expected	Got	
✓	19 34 72 113 210 82 103 243 48 222 86 166 2 161 213 246 167 58 219 131 114 64 47 80 135 236 206 242 71 211 106 83 251 200 42 177 231 103 146 238 45 241 202 91 140 204 64 135 194 135 134 250 81 251 226 19 177 18 61 48 10 247 94 10 139 30 13 204 235 144 21 221 249 110 52 122 233 104 63 98 137 91 163 181 167 67 222 156 61 101 251 111 186 113 23 231 114 150 66 30 33 94 143 187 64 226 60 206 196 209 206 243 64 97 0 252 58 60 227 162 30 139 116 165 6 125 205 78 108 138 50 86 116 168 31 214 53 12 168 97 97 137 191 74 7 226 175 14 114 59 125 157 42 89 202 81 6 109 58 253 148 110 196 3 116 164 147 21 221 230 55 58 13 190 97 103 120 226 43 76 106 180 120 226 132 166 49 70 14 198 19 109 104 74 31 25 234 9 131 204 157 74 52 23 177 57 41 168 163 225 217 250 174 217 69 35 50 37 104 195 57 88 220 143 241 185 240 155 47 103 98 185 237 216 247 219 53 208 250 82 157 154 60 180 87 190 76 188 69 121 48 71 48 38 134 67 147 129 114 63 73 113 79 102 163 166 59 228 177 179 212 129 40 97 23 177 161 11 136 135 171 59 75 52 24 8 19 188 106 4 78 144 151 133 51 115 237 173 211 58 195 4 220 122 8 181 56 228 213 16 96 27 132 24 20 191 252 88 106 213 79 204 106 149 194 126 179 252 206 202 86 112 170 183 254 8 8 111 193 0 59 108 145 244 46 243 67 226 214 158 189 10 95 101 231 188 82 66 217 216 238 121 94 74 241 199 138 132 46 160 189 68 81 119 116 2 129 127 111 213 241 43 227 165 186 187 167 38 230 226 14 188 118 243 102 210 184 249 38 140 188 64 48 34 23 94 128 122 138 152 45 94 183 192 1 134 13 63 181 34 199 16 68 169 152 228 46 49 158 101 187 246 225 188 250 75 72 196 61 145 55 97 176 102 221 218 223 254 33 168 163 87 148 41 189 104 45 178 108 103 143 170 162 130 224 39 184 33 162 70 204 242 85 241 224 164 188 220 220 66 17 112 107 38 185 125 128 131 126 192 109 236 49 123 163 87 28 240 250 167 15 30 62 240 166 215 121 1 178 63 112 61 6 73 245 7 220 208 1 177 230 76 114 9 205 247 143 45 71 182 37 73 192 229 16 100 124 94 28 215 145 168 138 243 86 121 55 178 6 21 49 234 183 162 251 118 37 24 119 68	0 0 0 0 0 243 0 0 0 0 0 0 0 246 0 0 0 0 0 0 0 0 0 0 0 242 0 0 0 0 251 0 0 0 0 0 0 0 0 241 0 0 0 0 0 0 0 0 250 0 251 0 0 0 0 0 0 0 247 0 0 0 0 0 0 0 0 0 0 249 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 251 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 243 0 0 0 252 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 253 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 250 0 0 0 0 0 0 0 0 0 0 0 241 0 240 0 0 0 0 0 0 247 0 0 0 250 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 252 0 0 0 0 0 0 0 0 0 0 252 0 0 0 0 0 0 254 0 0 0 0 0 0 0 0 244 0 243 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 241 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 241 0 0 0 0 0 0 0 0 0 0 0 243 0 0 0 249 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 246 0 0 250 0 0 0 0 0 0 0 0 0 0 0 0 0 254 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 242 0 241 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 240 250 0 0 0 0 240 0 0 0 0 0 0 0 0 0 0 245 0 0 0 0 0 0 0 0 0 247 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 243 0 0 0 0 0 0 0 0 0 251 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 243 0 0 0 0 0 0 0 246 0 0 0 0 0 0 0 0 0 0 0 242 0 0 0 0 251 0 0 0 0 0 0 0 0 241 0 0 0 0 0 0 0 0 0 250 0 251 0 0 0 0 0 0 0 247 0 0 0 0 0 0 0 0 0 0 249 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 251 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 243 0 0 0 252 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 253 0 252 0 0 0 0 0 0 0 0 0 0 0 0 252 0 0 0 0 0 254 0 0 0 0 0 0 0 0 244 0 243 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 241 0 0 0 0 0 0 0 0 0 0 0 0 0 0 241 0 0 0 0 0 0 0 0 0 0 0 0 243 0 0 0 249 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 246 0 0 250 0 0 0 0 0 0 0 0 0 0 0 0 254 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 242 0 241 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 240 250 0 0 0 0 240 0 0 0 0 0 0 0 0 0 0 245 0 0 0 0 0 0 0 0 0 247 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 243 0 0 0 0 0 0 0 0 0 251 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	✓

	Input	Expected	Got	
	0 113 180 32 171 149 188 228 22 213 52 126 37 227 221 25 69 204 113 7 143 27 195 94 138 43 172 166 4 73 205 236 216 54 70 186 220 86 150 241 44 180 53 122 67 227 49 167 237 254 146 144 28 200 192 168 120 4 183 83 200 74 180 108 79 160 41 250 137 113 116 73 159 187 43 248 76 97 183 45 226 236 40 177 77 79 165 7 240	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 241 0 0 0 0 0 0 0 0 0 254 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 250 0 0 0 0 0 0 0 248 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 241 0 0 0 0 0 0 0 0 0 254 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 250 0 0 0 0 0 0 0 248 0 0 0 0 0 0 0 0 0 0 0 0	

Passed all tests! ✓

Đúng

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

Câu hỏi 4

Đúng

Đạt điểm 10,00 trên 10,00

Một véc-tơ n chiều: $\vec{x} = (x_1, x_2, \ldots, x_n)$ có thể biểu diễn bằng một mảng gồm n số. Trong nhiều bài toán người ta muốn giá trị các chiều của véc-tơ nằm trong đoạn $[0, 1]$ (hoặc $[-1, 1]$) tức là $x_i \in [0, 1] \forall i$. Một cách để làm việc đó là chia các phần tử của mảng cho số lớn nhất có thể có của các phần tử đó.

Hàm `void normalize(double *out, int *in, int n)` nhận các tham số là:

- Con trỏ trỏ đến mảng đầu vào `in`. Mảng đầu vào chứa các số nguyên trong đoạn $[0, 255]$.
- Con trỏ trỏ đến mảng đầu ra `out`. Mảng đầu ra là mảng chuẩn hóa của mảng đầu vào, chứa các số thực sau khi chia số nguyên tương ứng của mảng đầu vào cho 255.
- Số nguyên `n` là số phần tử của hai mảng.

Nhiệm vụ của hàm `void normalize(double *out, int *in, int n)` là chuẩn hóa các giá trị trong mảng đầu vào `in` về khoảng $[0, 1]$ và lưu vào mảng đầu ra `out`.

Hãy viết mã C++ để hoàn thành hàm `void normalize(double *out, int *in, int n)` thực hiện các yêu cầu trên.

For example:

Input	Result
5 78 167 90 119 99	0.306 0.655 0.353 0.467 0.388

Answer: (penalty regime: 0 %)

```
void normalize(double *out, int *in, int n){
    for(int i=0; i<n; i++){
        out[i]=in[i]/255.0;
    }
}
```

	Input	Expected	Got	
✓	5 78 167 90 119 99	0.306 0.655 0.353 0.467 0.388	0.306 0.655 0.353 0.467 0.388	✓
✓	20 245 23 104 106 252 139 243 252 136 58 97 196 158 253 53 234 203 95 83 214	0.961 0.090 0.408 0.416 0.988 0.545 0.953 0.988 0.533 0.227 0.380 0.769 0.620 0.992 0.208 0.918 0.796 0.373 0.325 0.839	0.961 0.090 0.408 0.416 0.988 0.545 0.953 0.988 0.533 0.227 0.380 0.769 0.620 0.992 0.208 0.918 0.796 0.373 0.325 0.839	✓

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 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

```
1 void rFilter(char *s) {
```

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

Passed all tests! ✓

Đúng

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

Câu hỏi 6

Đúng

Đạt điểm 10,00 trên 10,00

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.

For example:

Input	Result
37 58	44 48 55

Answer: (penalty regime: 0 %)

```
bool check(int n){
    int temp=n;
    while(temp!=0){
        int digit=temp%10;
        if(digit==0||n%digit!=0){
            return false;
        }
        temp/=10;
    }
    return true;
}

int *selfDividingNumbers(int left, int right, int *returnSize){
    int cnt=0;
    for(int i=left; i<=right; i++){
        if(check(i)){
            cnt++;
        }
    }
    int isize=0;
    int *res=new int[cnt];
    for(int i=left; i<=right; i++){
        if(check(i)){
            res[isize]=i;
            isize++;
        }
    }
    *returnSize=cnt;
    return res;
}
```

	Input	Expected	Got	
✓	29 93	33 36 44 48 55 66 77 88	33 36 44 48 55 66 77 88	✓

	Input	Expected	Got	
✓	78 86			✓

Passed all tests! ✓

Đúng

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

Trở lại Khoá học