Question 1

Not answered

Mark 0.00 out of 10.00

[CountOddDigits]

Viết hàm đếm số lượng chữ số chẵn và số lượng chữ số lẻ của số nguyên dương n gồm k chữ số $(k < 20)$.

Đầu vào

Đầu vào từ bàn phím gồm $T + 1$ dòng.

- Dòng đầu tiên chứa số nguyên T là số lượng số cần kiểm tra;
- T dòng tiếp theo, mỗi dòng chứa một số nguyên n.

Đầu ra

In ra màn hình T dòng, mỗi dòng ghi ra số lượng chữ số chẵn và lẻ của n, cách nhau bởi một dấu cách.

For example:

Input	Result
1 12345	2 3

Answer: (penalty regime: 0 %)

```
1
```

Question 2

Not answered

Mark 0.00 out of 10.00

[CheckOddEven]

Viết hàm kiểm tra xem số nguyên dương n (gồm k chữ số - $k < 20$) có chỉ chứa các chữ số chẵn hoặc chỉ chứa các chữ số lẻ hay không.

Đầu vào

Đầu vào từ bàn phím gồm $T + 1$ dòng.

- Dòng đầu tiên chứa số nguyên T là số lượng số cần kiểm tra;
- T dòng tiếp theo, mỗi dòng chứa một số nguyên n.

Đầu ra

In ra màn hình T dòng, mỗi dòng ghi ra "yes" hoặc "no" tương ứng với mỗi số n có thỏa mãn điều kiện không.

For example:

Input	Result
3 123214 1353 2248	no yes yes

Answer: (penalty regime: 0 %)

```
1
```

Question 3

Not answered

Mark 0.00 out of 10.00

[MaxOddDivisor]

Viết hàm tính ước số lẻ lớn nhất khác chính nó của một số nguyên dương n cho trước.

Đầu vào

Đầu vào từ bàn phím gồm $T + 1$ dòng.

- Dòng đầu tiên chứa số nguyên T là số lượng số cần kiểm tra;
- T dòng tiếp theo, mỗi dòng chứa một số nguyên n $(n < 100)$.

Đầu ra

In ra màn hình T dòng, mỗi dòng ghi ra ước số lẻ lớn nhất khác chính nó của n.

For example:

Input	Result
1 6	3

Answer: (penalty regime: 0 %)

```
1
```

Question 4

Not answered

Mark 0.00 out of 10.00

[PrimeDigits]

Viết hàm đếm số lượng chữ số nguyên tố của số nguyên dương n gồm k chữ số $(k < 100)$.

Đầu vào

Đầu vào từ bàn phím gồm $T + 1$ dòng.

- Dòng đầu tiên chứa số nguyên T là số lượng số cần kiểm tra;
- T dòng tiếp theo, mỗi dòng chứa một số nguyên n.

Đầu ra

In ra màn hình T dòng, mỗi dòng ghi ra số lượng chữ số nguyên tố của n.

For example:

Input	Result
1 12345	3

Answer: (penalty regime: 0 %)

```
1
```

Question 5

Not answered

Mark 0.00 out of 10.00

[Circle]

Ban đầu, con rô-bốt ở vị trí (0, 0). Cho một chuỗi di chuyển của rô-bốt, đánh giá xem rô-bốt có tạo thành một chu trình hay không, nói cách khác là nó có trở lại vị trí ban đầu hay không.

Chuỗi di chuyển được biểu diễn bởi một xâu kí tự. Mỗi bước di chuyển được biểu diễn bởi một kí tự, và giả sử bước của rô-bốt là như nhau. Các bước di chuyển của rô-bốt là `R` (Phải), `L`(Trái), `U` (Lên) and `D` (Xuống). Viết hàm `bool circle(string moves)` trả về $true$ nếu chuỗi di chuyển tạo thành chu trình, ngược lại $false$.

For example:

Test	Input	Result

`string moves;`
`cin >> moves;`
`cout << circle(moves);`
	`LR`	`1`

`string moves;`
`cin >> moves;`
`cout << circle(moves);`
	`LL`	`0`

Answer: (penalty regime: 0 %)

```
1
```

Question 6

Not answered

Mark 0.00 out of 10.00

[CollinearPoints]

Viết hàm `bool isCollinear(int x1, int y1, int x2, int y2, int x3, int y3)` kiểm tra xem 3 điểm trong hệ tọa độ 2 chiều có thẳng hàng hay không, các tham số đầu vào lần lượt là tọa độ của các điểm.

Chú ý: Bạn chỉ phải viết đúng hàm theo yêu cầu. Hàm ***main()*** và thư viện đã có sẵn.

For example:

Test	Input	Result

`int x1, x2, x3, y1, y2, y3;`
`cin >> x1 >> y1 >> x2 >> y2 >> x3 >> y3;`
`if(isCollinear(x1, y1, x2, y2, x3, y3)){`
`  cout << "Collinear.";`
`} else {`
`  cout << "Not collinear.";`
`}`
	`-15 18 45 46 36 -59`	`Not collinear.`

Answer: (penalty regime: 0 %)

```
1
```

Question 7

Not answered

Mark 0.00 out of 10.00

[Floyd'sTriangle]

Viết hàm `void printFloydsTriangle(int nRows)` để in tam giác Floyd có số hàng là $nRows$, các số cách nhau bởi một dấu cách.

Lưu ý: bắt buộc phải viết hàm yêu cầu trong đề bài, không cần viết hàm `main()`, không cần thêm các dòng `#include` và có thể viết các hàm thêm tùy ý.

For example:

Input	Result
4	1 2 3 4 5 6 7 8 9 10

Answer: (penalty regime: 0 %)

```
1
```

Question 8

Not answered

Mark 0.00 out of 10.00

[IsomorphicWords]

Hai từ được gọi là đẳng cấu nếu các kí tự trong một từ có thể được ánh xạ để được từ còn lại. Ánh xạ một kí tự nghĩa là thay đổi toàn bộ các kí tự đó trong từ bởi một kí tự khác. Thứ tự của các kí tự được giữ không đổi. Hai kí tự khác nhau không thể ánh xạ đến cùng một kí tự, nhưng một kí tự có thể ánh xạ thành chính nó. Ví dụ, từ "abca" và "zbxz" là hai từ đẳng cấu vì ta có thể ánh xạ 'a' thành 'z', 'b' thành 'b' và 'c' thành 'x'.

Viết hàm `bool isIsomorphic(string a, string b)` trả về $true$ nếu hai từ a và b là đẳng cấu, ngược lại $false$. Bạn có thể giả sử hai từ đầu vào có cùng độ dài.

For example:

Test	Input	Result

`string a, b;`
`cin >> a >> b;`
`cout << isIsomorphic(a, b);`
	egg
	add
		1

Answer: (penalty regime: 0 %)

```
1
```

Question 9

Not answered

Mark 0.00 out of 10.00

[TriangleTypes]

Viết hàm `int getTriangleType(int a, int b, int c)` nhận 3 tham số là độ dài ba cạnh (không chắc là tam giác) và trả về kiểu của tam giác. 0 nếu không phải tam giác, 1 nếu là tam giác cân, 2 là tam giác đều và 3 là tam giác thường.

For example:

Input	Result
3 3 3	2

Answer: (penalty regime: 0 %)

```
1
```

Question 10

Not answered

Mark 0.00 out of 10.00

[UglyNumber]

Số xấu xí là một số nguyên dương mà các thừa số nguyên tố của nó chỉ bao gồm $2, 3, 5$. Ví dụ, $6, 8$ là xấu xí trong khi đó 14 không xấu xí vì nó có một thừa số nguyên tố là 7. 1 cũng được coi là một số xấu xí.

Viết hàm `bool isUgly(int n)` trả về $true$ nếu số n là xấu xí, ngược lại trả về $false$.

For example:

Test	Input	Result

`int n;`
`cin >> n;`
`cout << isUgly(n);`
	25	1

Answer: (penalty regime: 0 %)

```
1
```

Back to Course