Divisores próprios

Requested files: user.c, input.txt (Download)
Type of work: Individual work
Grade settings: Maximum grade: 10 **Hidden**
Run: Yes **Evaluate**: Yes **Evaluate just on submission**: Yes
Automatic grade: Yes

Divisores próprios de um número positivo n são todos os divisores inteiros positivos de *n*, exceto o próprio *n*. Por exemplo, os divisores próprios do número *n*=6 são 1, 2 e 3; para *n*=30, os divisores próprios são 1, 2, 3, 5, 6, 10 e 15.

Escreva um programa para imprimir os divisores próprios de um número *n*.

Dicas:

- O operador de resto em C é %, assim para calcular o resto de um número *x* entre 2 na variável *resto*, deve ser escrita a linha:
  `resto = x % 2;`

Entrada e Saída:

A entrada será constituída por um número inteiro positivo *n* que é maior do que 1. Cada linha no arquivo "input.txt" representará uma entrada para o programa. Como saída você deve imprimir a lista dos divisores próprios do número *n*, separados pela quebra de linha.

Exemplos de entrada	**Saída para os exemplos de entrada**
6	1 2 3
30	1 2 3 5 6 10 15
16	1 2 4 8

45	1 3 5 9 15

Requested files

user.c

```
 1 #include <stdio.h>
 2
 3 int main() {
 4           int n;
 5           scanf("%d",&n);
 6           // escreva seu código aqui
 7
 8           return 0;
 9 }
10
```

input.txt

```
1 6
2 30
3 16
4 45
```

Execution files

vpl_run.sh

```
#! /bin/bash

cat > vpl_execution <<EEOOFF
#! /bin/bash

prog1="user"
prog2="test"
gcc \${prog1}.c -o \${prog1} -lm | grep -v Note > grepLines.out
gcc \${prog2}.c -o \${prog2} -lm | grep -v Note > grepLines.out
if [ -s grepLines.out ] ; then
     echo "Some compiler ERRORS reported"
     cat grepLines.out
     exit
fi

while IFS='' read -r line || [[ -n "\${line}" ]]; do
    if [[ !  -z  "\${line}"  ]]; then
        echo "\${line}" > in.txt
        echo "----------Your program----------"

        ./user < in.txt
        echo "--------------------"
        echo "----------Test program----------"
        ./test < in.txt
        echo "--------------------"
    fi
done < input.txt
EEOOFF

chmod +x vpl_execution
```

vpl_debug.sh

```
```

vpl_evaluate.sh

```
#!/bin/bash

cat >vpl_execution << 'EOF'
#!/bin/bash
user="user"
test="test"
params_file="params.in"

# > Compile the executable
gcc $user.c -o $user -lm
gcc $test.c -o $test -lm

# > The params file format:
# .. First line: number of tests;
# .. Second line: number of inputs from each test
typeset -i num_tests=$(head -n 1 $params_file)
typeset -i num_input=$(tail -n 1 $params_file)
declare -i successes=0

# > Read every test ...
for num in $(seq $num_input $(($num_tests+$num_input-1)));
do
    echo $num
    echo $num > "in.txt"
    # > Execute both user and test programs with the same input
    echo `./$user < in.txt` > ${user}_out
    echo `./$test < in.txt` > ${test}_out

    diff -y -w -B --ignore-all-space ${user}_out ${test}_out > diff.out
    # > Wrong answer
    if (($? > 0)); then
        echo "Comment :=>> Incorrect output found on n = $num"
        echo "Comment :=>>- Your output"
        echo "<|--"
        cat ${user}_out
        echo "--|>"
        echo ""
        echo "Comment :=>>- Expected output "
        echo "<|--"
        cat ${test}_out
        echo "--|>"
    # > Right answer
    else
        successes=$((successes+1))
        #echo "Comment :=>> Correct output."
    fi
done
echo "------------------------------"
echo "Comment :=>>- Your success rate is ${successes}/${num_tests}."
echo "Grade :=>>$(((10*successes)/num_tests))"
echo "------------------------------"

EOF

chmod +x vpl_execution
```

vpl_evaluate.cases

test.c

```c
#include <stdio.h>

int main() {
    int n;
    scanf("%d",&n);
    int k;
    for (k = 1 ; k <= n/2 ; k++) {
        if (n % k == 0) printf("%d\n", k);
          }
    return 0;
}
```

params.in

```
100
2
```

VPL 3.1.4
