Sequência inversa Fibonacci de base n e m

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

A sequência de números Fibonacci é a sequência de números inteiros positivos na qual cada termo subsequente corresponde à soma dos dois números anteriores. Normalmente, a sequência de Fibonacci é definida inicialmente pelos números *n*=1 e *m*=1. Entretanto, ela pode iniciar com quaisquer outros dois valores inteiros positivos para *n* e *m*.

Definimos a sequência inversa Fibonacci de base *n* e *m*, denotada por "**inv($F_{n,m}$)**" como a sequência de números inteiros positivos na qual cada termo sub-sequente corresponde à diferença dos dois números anteriores. Por exemplo, a sequência inversa Fibonacci de base 81 e 50 "**inv($F_{81,50}$)**" é a sequência de números:

81, 50, 31, 19, 12, 7, 5, 2, 3

A sequência inversa Fibonacci é iniciada pelos números *n* e *m*, sendo *n* sempre maior do que *m* (*n*>*m*) e ela termina quando a diferença dos dois números anteriores *n* e *m* é menor ou igual do que 0. No exemplo acima, termina quando 2-3=-1.
Escreva um programa usando a linguagem de programação C, que dados os valores *n* e *m*, imprima a sequência inversa Fibonacci de base *n* e *m*.

Entrada e Saída:

A entrada será constituída por pares de números inteiros *n* e *m* separados por um espaço em branco. Todos os inteiros serão números maiores do que 0, assim como sempre *n* será maior do que *m*. Cada linha no arquivo "input.txt" representará uma entrada para o programa. Como saída você deve imprimir a sequência inversa Fibonacci de base *n* e *m* (incluindo os números *n* e *m*).

Exemplos de entrada	**Saída para os exemplos de entrada**
81 50	81 50 31 19 12 7 5 2 3
34 21	34 21 13 8 5 3 2 1 1

1189 360	1189 360 829

Requested files

user.c

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

input.txt

```
1 34 21
2 1189 360
3 832040 514229
4 34523412 31523412
```

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 "ERROS no compilador"
     cat grepLines.out
     exit
fi

while IFS='' read -r line || [[ -n "\${line}" ]]; do
    if [[ !  -z  "\${line}"  ]]; then
        echo "\${line}" > in.txt
        echo -e "----------Saída de seu programa----------"
        echo -e "Para a entrada: \${line}"
        echo -e "-----------------------------------------"
        ./user < in.txt
        echo -e "\n"

        echo -e "----------Saída esperada do programa----------"
        echo -e "Para a entrada: \${line}"
        echo -e "-----------------------------------------------"
        ./test < in.txt
        echo -e "\n"
    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"
    input_tests="tests.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 0 $(($num_tests-1)));
    do
        > "in.txt"
        # ... param by param, composing an input file
        for input in $(seq 0 $(($num_input-1)));
        do
            typeset -i selected_line=$(((num*num_input)+input+1))
            cmd="$selected_line!d"
            sed $cmd $input_tests >> "in.txt" # get the selected input from file
        done

        # > 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 test $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

```
#include <stdio.h>

int main() {
  int n, m;
  scanf("%d %d", &n, &m);
  printf("%d\n%d\n", n, m);
  while (n-m>0) {
      int m_temp = m;
      m = n - m;
      n = m_temp;
      printf("%d\n", m);
  }
  return 0;
}

```

params.in

```
10
2
```

tests.in

```
34
21
1189
360
832040
514229
34523412
31523412
514229
28657
2189
660
54553312
21583412
614229
38657
2380
760
23890
6770
```

VPL 3.1.4