O i-ésimo prêmio da caça-níquel

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

Para cada rodada *i*, uma caça-níquel tem o seguinte programa de premiação:

- Se é apostado um número *n* ímpar de moedas, elas serão duplicadas;
- Se a quantidade de moedas *n* é um número par, então:
  - O valor total obtido do prêmio será a diferença absoluta dos números pares e ímpares na sequência inversa Fibonacci de base *n* e *n-i* "inv($F_{n,n-i}$)" se *n-i* é maior do que 0;
  - Caso contrario, se *n-i* é menor ou igual do que 0, o valor do prêmio será a metade das moedas.

Nas seguintes rodadas, o comportamento de premiação será repetido com *n* sendo todo o montante acumulado de moedas. No entanto, se *n* é um múltiplo de 5 ou ele for maior do que 10000 moedas na rodada, todo o montante será perdido na próxima rodada. Assim, para uma aposta inicial de *n*=20, a seguinte sequência de premiação da caça níquel será obtida: 18, 50, 0, 0, ...

Escreva um programa, que dado dois números inteiros *n* e *i*, determine o prêmio da i-ésima rodada na caça níquel para uma aposta inicial com *n* moedas. No máximo, podem ser efeituadas 100 rodadas por jogo (*i*<=100). Assim, para *n*=13 e *i*=3, o prêmio da i-ésima rodada será 68.
Detalhando a sequência de premiação para *n*=13: 26, 74, 68, ...

- Na rodada *i*=1 com a aposta *n*=13, o prêmio é 26=13*2
- Na rodada *i*=2 com a aposta *n*=26, o prêmio é 74 devido a que a sequência inversa Fibonacci para *n*=26 e *m*=24 "inv($F_{26,24}$)" é 26, 24, 2, 22, assim 74=26+24+2+22.
- Na rodada *i*=3 com a aposta *n*=74, o prêmio é 68 devido a que a sequência inversa Fibonacci para n=74 e m=71 "inv($F_{74,71}$)" é 74, 71, 3, 68, assim 68=(74+68)-(71+3).
- ...

Dicas:

- A sequência inversa Fibonacci de base *n* e *m* "inv($F_{n,m}$)" é a sequência de números inteiros positivos na qual cada termo sub-sequente corresponde à diferença dos dois números anteriores. Por exemplo, para *n*=81 e *m*=50, a sequência inversa Fibonacci é 81, 50, 31, 19, 12, 7, 5, 2, 3.
- Para calcular o valor absoluto de um número *x*, uma solução básica é multiplicar o número por -1 se *x* é menor do que 0 (zero).

Entrada e Saída:

A entrada será constituída por dois números inteiros *n* e *i*, *n* maior do que 0 e menores a 10000, *i* maior do que 0 e menor ou igual a 100. Cada linha no arquivo "input.txt" representará uma entrada para o programa.

Como saída, você deve imprimir três números, o valor de *n*, *i* e o prêmio da i-ésima rodada na máquina caça níquel. Veja abaixo alguns exemplos de entrada/saída:

Exemplos de entrada	**Saída para os exemplos de entrada**
13 5	13 5 186

20 8	20 8 0
18 2	18 2 44
23 3	23 3 128
346 3	346 3 1022

Requested files

user.c

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

input.txt

```
1 15 5
2 22 8
3 30 10
4 37 25
5 99 13
```

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

```c
#include <stdio.h>

int invFib(int n, int m) {
    int accum=0;
    if (n%2==0) accum+=n; else accum-=n;
  if (m%2==0) accum+=m; else accum-=m;
  while (n-m>0) {
      int m_temp = m;
      m = n - m;
      n = m_temp;
      if (m%2==0) accum+=m; else accum-=m;
  }
  if (accum<0) accum *= -1;
  return accum;
}

int main() {
  int i, n;
  scanf("%d %d", &n, &i);
  int k=0, accum = n;

  if (i>100) { return -1; }

  do {
      k++;
          if (accum % 2 != 0) {
              accum *= 2;
          } else {
              if (accum-k>0) {
                accum = invFib(accum, accum-k);
            } else {
                accum = accum/2;
            }
        }
    } while (k<i && (accum % 5 != 0) && accum<=10000);

    if (k<i && ((accum % 5 == 0) || accum>10000)) {
        accum = 0;
    }

  printf("%d %d %d\n", n, i, accum);
  return 0;
}
```

params.in

```
17
2
```

tests.in

```
 1 1
 2 1
 3 13
 4 5
 5 20
 6 8
 7 18
 8 2
 9 23
10 3
11 346
12 3
13 9999
14 100
15 4999
16 2
17 23
18 6
19 23
20 8
21 17
22 7
23 17
24 8
25 17
26 9
27 13
28 8
29 13
30 9
31 456
32 12
33 145
34 3
```

VPL 3.1.4
