Ganhando o prêmio maior 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*=13, a seguinte sequência de premiação da caça níquel será obtida:

26, 74, 68, 196, 186, 546, 532, 1580,0, 0, ...

Escreva um programa, que dado dois números inteiros *min* e *max*, os quais representam respectivamente o mínimo e máximo de moedas que podem ser apostadas no início do jogo, determine: o valor de *n* no intervalo de *min* e *max* (incluindo *min* e *max*) que devemos apostar no início do jogo para ganhar o prêmio maior, o número *i* de iterações (rodadas) necessárias para ganhar esse prêmio e o valor do prêmio maior. No máximo podem ser feitas 100 rodadas por jogo (*i*<=100), a rodada 101 não existe.

Importante:

- O prêmio maior é o máximo beneficio que podemos sacar da caça-níquel com o programa de premiação detalhado acima.

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 *min* e *max*, ambos maiores do que 0 e menores que 10000. A entrada *min* será sempre menor do que *max* e cada linha no arquivo "input.txt" representará uma entrada para o programa.

Como saída, você deve imprimir cinco números, os valor de *min*, o valor de *max*, o valor *n* a ser apostado para ganhar o prêmio maior, a iteração na qual devemos parar para ganhar o prêmio maior e o valor do prêmio maior. Veja abaixo alguns exemplos de entrada/saída:

Exemplos de entrada	**Saída para os exemplos de entrada**
5 13	5 13 13 8 1580

8 20	8 20 17 8 2228
2 18	2 18 17 8 2228
3 23	3 23 23 8 3200
1 9999	1 9999 4999 2 29990

Requested files

user.c

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

input.txt

```
1 5 13
2 8 20
3 2 18
4 3 23
5 3 346
```

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 min, max, n, i, N, I, prize, maxPrize=0;
  scanf("%d %d", &min, &max);

  for (n=min; n<=max; n++) {
      i = 0;
      prize = n;
      do {
          i++;
          if (prize%2 != 0) {
              prize *= 2;
          } else {
              if (prize-i>0) {
                  prize = invFib(prize, prize-i);
              } else {
                  prize = prize/2;
              }
          }

          if (maxPrize<prize) {
              N = n;
              I = i;
              maxPrize = prize;
          }
      } while(prize%5!=0 && prize<=10000 && i < 100);
  }

  printf("%d %d %d %d %d\n", min, max, N, I, maxPrize);
  return 0;
}

```

params.in

```
17
2
```

tests.in

```
 1 1
 2 2
 3 1
 4 9999
 5 9
 6 99
 7 5
 8 13
 9 2
10 20
11 3
12 23
13 5
14 13
15 8
16 20
17 10
18 35
19 25
20 37
21 99
22 999
23 20
24 130
25 35
26 100
27 34
28 125
29 37
30 253
31 99
32 136
33 34
34 349
```

VPL 3.1.4
