1. Programa en prolog (archivo: clasificaNumero.pl)

lista([20,22,31,37,7,9]).

while([],I,I).

while([X|Res],I,F):-Aux is F mod I, Aux =:= 0, X is I,

Aux2 is I+1, while( Res,Aux2,F ).

while(L,I,F):- Aux is I+1, while(L,Aux,F).

divisores(Num,L):- while(L,1,Num).

%Rertorna en forma de Lista todos los Divisores de Num

sumaLista([],0).

sumaLista([X|Res],C):-sumaLista(Res,C1), C is C1+X.

%Dada una Lista, retorna en C la suma numerica de cada Elemento

comparaTipo(Num,SumDiv,X):-Num > SumDiv, X = "a".

comparaTipo(Num,SumDiv,X):-Num = SumDiv, X = "b".

comparaTipo(Num,SumDiv,X):-Num < SumDiv, X = "c".

%Dado un numero y la suma de sus divisores, terna en X el tipo

for(I,I,[]).

for(I,Para,Lista):-

divisores(I,Div),

sumaLista(Div,sumDiv),

comparaTipo(O,sumDiv,Lt),

append(Lista,Lt,ListaNueva)

I is I +1,

for(I,Para,ListaNueva).

clasifica(20,L):-for(1,20,L).

1. Multiplicación binaria en Maquina de Turing (archivo: MT.txt)

name: Multiplicacion Binaria

init: qInicio

accept: qFin

qInicio,0,\_,\_

qInicio,0,\_,\_,>,-,-

qInicio,1,\_,\_

qInicio,1,\_,\_,>,-,-

qInicio,\*,\_,\_

qCopiar,\_,\_,\_,>,-,-

qCopiar,0,\_,\_

qCopiar,\_,0,\_,>,>,-

qCopiar,1,\_,\_

qCopiar,\_,1,\_,>,>,-

qCopiar,\_,\_,\_

qIzquierda,\_,\_,\_,<,-,-

qIzquierda,\_,\_,\_

qIzquierda,\_,\_,\_,<,-,-

qIzquierda,0,\_,\_

qAnaliza,0,\_,\_,-,-,-

qIzquierda,1,\_,\_

qAnaliza,1,\_,\_,-,-,-

qAnaliza,0,\_,\_

qBanda2,0,\_,\_,-,-,-

qAnaliza,1,\_,\_

qSum,1,\_,\_,-,<,<

qAnaliza,\_,\_,\_

qBorrar,\_,\_,\_,>,-,-

qBanda2,0,\_,\_

qAnaliza,0,0,\_,<,>,-

qBanda2,1,\_,\_

qAnaliza,1,0,\_,<,>,-

qSum,1,0,0

qSum,1,0,0,-,<,<

qSum,1,0,1

qSum,1,0,1,-,<,<

qSum,1,1,0

qSum,1,1,1,-,<,<

qSum,1,1,1

qSumA,1,1,0,-,<,<

qSum,1,0,\_

qSum,1,0,0,-,<,<

qSum,1,1,\_

qSum,1,1,1,-,<,<

qSum,1,\_,0

qSum,1,\_,0,-,<,<

qSum,1,\_,1

qSum,1,\_,1,-,<,<

qSum,1,\_,\_

qSumB,1,\_,\_,-,>,>

qSumA,1,1,1

qSumA,1,1,1,-,<,<

qSumA,1,1,0

qSumA,1,1,0,-,<,<

qSumA,1,0,1

qSumA,1,0,0,-,<,<

qSumA,1,0,0

qSum,1,0,1,-,<,<

qSumA,1,0,\_

qSum,1,0,1,-,<,<

qSumA,1,1,\_

qSumA,1,1,0,-,<,<

qSumA,1,\_,0

qSum,1,\_,1,-,<,<

qSumA,1,\_,1

qSumA,1,\_,0,-,<,<

qSumA,1,\_,\_

qSum,1,\_,1,-,<,<

qSumB,1,\_,0

qSumB,1,\_,0,-,>,>

qSumB,1,\_,1

qSumB,1,\_,1,-,>,>

qSumB,1,0,\_

qSumB,1,0,\_,-,>,>

qSumB,1,1,\_

qSumB,1,1,\_,-,>,>

qSumB,1,0,0

qSumB,1,0,0,-,>,>

qSumB,1,0,1

qSumB,1,0,1,-,>,>

qSumB,1,1,0

qSumB,1,1,0,-,>,>

qSumB,1,1,1

qSumB,1,1,1,-,>,>

qSumB,1,\_,\_

qBanda2,1,\_,\_,-,-,-

qBorrar,0,\_,\_

qBorrar,\_,\_,\_,>,-,-

qBorrar,1,\_,\_

qBorrar,\_,\_,\_,>,-,-

qBorrar,\_,\_,\_

qCopiaF,\_,\_,\_,<,<,<

qCopiaF,0,0,0

qCopiaF,0,\_,\_,<,<,<

qCopiaF,0,0,1

qCopiaF,1,\_,\_,<,<,<

qCopiaF,0,1,0

qCopiaF,0,\_,\_,<,<,<

qCopiaF,0,1,1

qCopiaF,1,\_,\_,<,<,<

qCopiaF,1,0,0

qCopiaF,0,\_,\_,<,<,<

qCopiaF,1,0,1

qCopiaF,1,\_,\_,<,<,<

qCopiaF,1,1,0

qCopiaF,0,\_,\_,<,<,<

qCopiaF,1,1,1

qCopiaF,1,\_,\_,<,<,<

qCopiaF,0,\_,\_

qCopiaF,\_,\_,\_,<,-,-

qCopiaF,1,\_,\_

qCopiaF,\_,\_,\_,<,-,-

qCopiaF,\_,0,\_

qCopiaF,\_,\_,\_,-,<,-

qCopiaF,\_,1,\_

qCopiaF,\_,\_,\_,-,<,-

qCopiaF,\_,\_,0

qCopiaF,0,\_,\_,<,-,<

qCopiaF,\_,\_,1

qCopiaF,1,\_,\_,<,-,<

qCopiaF,0,0,\_

qCopiaF,\_,\_,\_,<,<,-

qCopiaF,0,1,\_

qCopiaF,\_,\_,\_,<,<,-

qCopiaF,1,0,\_

qCopiaF,\_,\_,\_,<,<,-

qCopiaF,1,1,\_

qCopiaF,\_,\_,\_,<,<,-

qCopiaF,0,\_,0

qCopiaF,0,\_,\_,<,<,<

qCopiaF,1,\_,0

qCopiaF,0,\_,\_,<,<,<

qCopiaF,0,\_,0

qCopiaF,0,\_,\_,<,<,<

qCopiaF,0,\_,1

qCopiaF,1,\_,\_,<,<,<

qCopiaF,1,\_,1

qCopiaF,1,\_,\_,<,<,<

qCopiaF,\_,0,0

qCopiaF,0,\_,\_,<,<,<

qCopiaF,\_,0,1

qCopiaF,1,\_,\_,<,<,<

qCopiaF,\_,1,0

qCopiaF,0,\_,\_,<,<,<

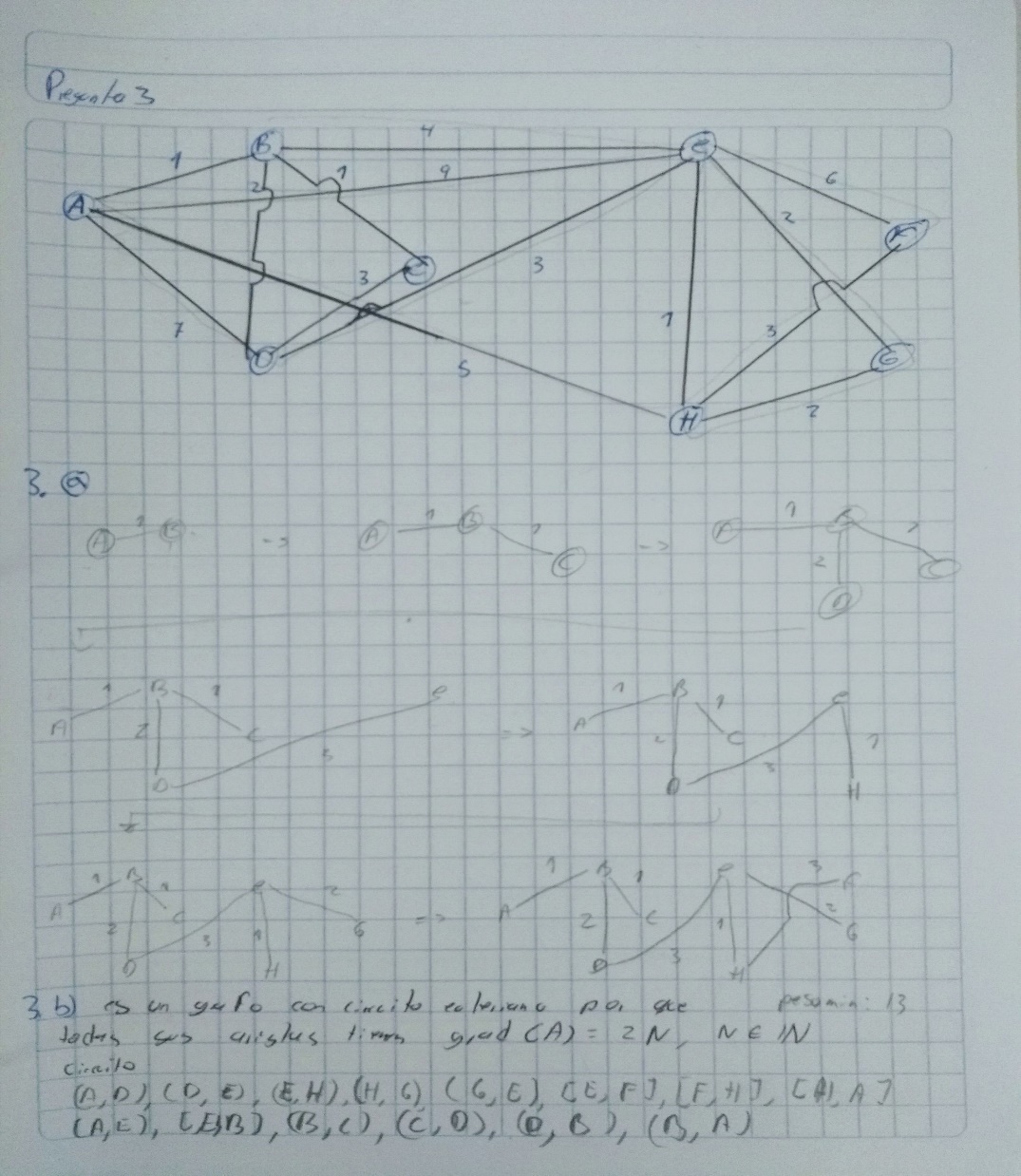
qCopiaF,\_,1,1

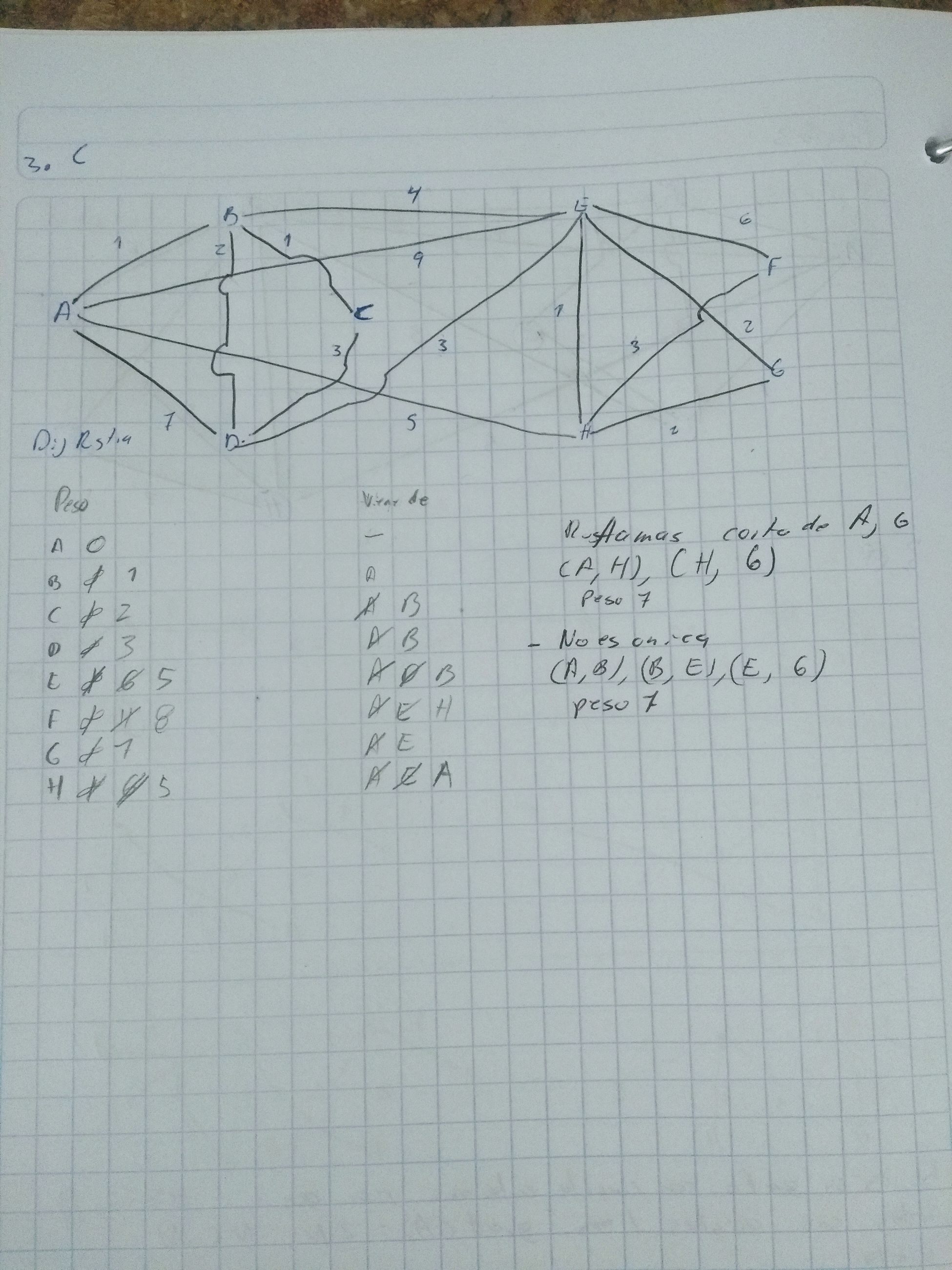
qCopiaF,1,\_,\_,<,<,<

qCopiaF,\_,\_,\_

qFin,\_,\_,\_,-,-,-

1. Grafos





1. Algoritmos recursivos e iterativos X ala N (archivos: exponencialRecursivo.cpp y exponencialIterativo.cpp)

#include <iostream>

using namespace std;

double potencia(int, int);

int main(){

int x, y;

cout << "Introduzca base: ";

cin >> x;

do{

cout << "Introduzca exponente >= 0 : ";

cin >> y;

}while(y < 0);

cout << x << " elevado a " << y << " = " << potencia(x,y) << endl;

}

double potencia(int x, int y){

double resultado = 1;

int i;

for(i = 1;i <= y;i++)

resultado = resultado \* x;

return resultado;

}

#include <iostream>

using namespace std;

double potencia(int, int);

int main(){

int x, y;

cout << "Introduzca base: ";

cin >> x;

do{

cout << "Introduzca exponente >=0 : ";

cin >> y;

}while(y < 0);

cout << x << " elevado a " << y << " = " << potencia(x,y) << endl;

}

double potencia(int x, int y){

if (y == 0)

return 1;

else

return x\*(potencia(x,y-1));

}