let u0 = 241 ;;

(\*Q1\*)

let create\_tab n =

let tab = Array.make (n+1) u0 in

for i = 1 to n do

tab.(i) <- 15091 \* tab.(i-1) mod 64007

done;

tab

;;

(\*

let q1\_a = (create\_tab 17).(17) ;;

let q1\_b = (create\_tab 42).(42) ;;

let q1\_c = (create\_tab 8000).(8000) ;;

\*)

(\*Q2\*)

type sign =

|Plus of int

|Minus of int

;;

let taille (p,q,r) =

let tab = create\_tab (q\*r+r) in

let clause = Array.make q [] in

let m = ref 0 in

for j = 0 to q-1 do

for k = 0 to r-1 do

begin

let a = tab.(j\*r+k) mod p in

if tab.(j\*r+k) < 32003 then

begin

if not(List.mem (Plus(a)) clause.(j)) then

clause.(j) <- Plus(a)::clause.(j)

end

else

begin

if not(List.mem (Minus(a)) clause.(j)) then

clause.(j) <- Minus(a)::clause.(j)

end

end

done;

done;

for i = 0 to q-1 do

m := (List.length clause.(i)) + !m

done;

!m,clause

;;

(\*

let q2\_a = taille (4,15,2) ;;

let q2\_b = taille (3,30,3) ;;

let q2\_c = taille (8,60,3) ;;

let q2\_d = taille (10,100,4) ;;

\*)

(\*Q3\*)

(\*Utile pour remove un element de la liste\*)

let same\_var a b =

match a,b with

|Plus(x),Plus(y) ->

if x = y then true

else false

|Minus(x),Minus(y) ->

if x = y then true

else false

|\_ -> false

;;

let rec remove\_var l a =

match l with

|[] -> l

|x::xs ->

if same\_var x a then

xs

else

x::(remove\_var xs a)

;;

let nb\_clauses\_trivial (p,q,r) =

let (\_,tab) = taille (p,q,r) in

let count = ref 0 in

let rec explore l n =

if n = 0 then l

else

match l with

|[] -> l

|Plus(x)::xs ->

if List.mem (Minus(x)) xs then

begin

incr count;

remove\_var(remove\_var xs (Plus(x))) (Minus(x))

end

else

explore (xs@[Plus(x)]) (n-1)

|Minus(x)::xs ->

if List.mem (Plus(x)) xs then

begin

incr count;

remove\_var(remove\_var xs (Minus(x))) (Plus(x))

end

else

explore (xs@[Minus(x)]) (n-1)

in

for i = 0 to q-1 do

match tab.(i) with

|[] -> ()

|\_ ->

let n = List.length tab.(i) in

tab.(i) <- explore tab.(i) n

done;

!count,tab

;;

let q3\_a = nb\_clauses\_trivial (4,15,2) ;;

let q3\_b = nb\_clauses\_trivial (3,30,3) ;;

let q3\_c = nb\_clauses\_trivial (8,60,3) ;;

let q3\_d = nb\_clauses\_trivial (10,100,4) ;;

(\*Q4\*)

let var a =

match a with

|Plus(x) -> x

|Minus(x) -> x

;;

let nb\_variables\_forces (p,q,r) =

let (\_,tab) = nb\_clauses\_trivial (p,q,r) in

let valeurs\_forces = ref [] in

let rec find\_valeurs\_forces l =

match l with

|x::xs ->

begin

match xs with

|[] ->

begin

if not(List.mem x !valeurs\_forces) then

valeurs\_forces := !valeurs\_forces@[x]

end;

|y::ys -> ()

end

|\_ -> ()

in

for i = 0 to q-1 do

match tab.(i) with

|[] -> ()

|\_ ->

find\_valeurs\_forces tab.(i)

done;

List.length (!valeurs\_forces), !valeurs\_forces

;;

let q4\_a = nb\_variables\_forces (4,15,2) ;;

let q4\_b = nb\_variables\_forces (3,30,3) ;;

let q4\_c = nb\_variables\_forces (8,60,3) ;;

let q4\_d = nb\_variables\_forces (10,100,4) ;;

(\*Q5\*)

exception Novalforce ;;

let taille\_apres\_une\_valeur\_force (p,q,r) =

let (\_,tab) = nb\_clauses\_trivial (p,q,r) in

let (\_,valeurs\_forces) = nb\_variables\_forces (p,q,r) in

let m = ref 0 in

let first\_val\_force l =

match l with

|[] -> raise Novalforce

|x::xs -> x

in

let rec replace\_neg l a =

match l with

|[] -> []

|x::xs ->

if a = x then

xs

else

x::(replace\_neg xs a)

in

let rec replace\_val l1 l2 a =

match l1 with

|[] -> l2

|x::xs ->

if x = a then

[]

else

(replace\_val xs (l2@[x]) a)

in

let reverse a =

match a with

|Plus(x) -> Minus(x)

|Minus(x) -> Plus(x)

in

try

match (first\_val\_force valeurs\_forces) with

|Plus(x) ->

let a = (first\_val\_force valeurs\_forces) in

for i = 0 to q-1 do

begin

tab.(i) <- replace\_val tab.(i) [] a ;

tab.(i) <- replace\_neg tab.(i) (reverse a)

end

done;

for j = 0 to q-1 do

m:= List.length(tab.(j))+ !m

done;

!m

|Minus(x) ->

let a = (first\_val\_force valeurs\_forces) in

for i = 0 to q-1 do

begin

tab.(i) <- replace\_val tab.(i) [] (reverse a) ;

tab.(i) <- replace\_neg tab.(i) a

end

done;

for j = 0 to q-1 do

m:= List.length(tab.(j))+ !m

done;

!m

with

|Novalforce ->

for j = 0 to q-1 do

m:= List.length(tab.(j))+ !m

done;

!m

;;

(\*Q6\*)

let taille\_after\_all\_valeurs\_forces (p,q,r) =

let (\_,tab) = nb\_clauses\_trivial (p,q,r) in

let (\_,valeurs\_forces) = nb\_variables\_forces (p,q,r) in

let m = ref 0 in

let rec replace\_neg l a =

match l with

|[] -> []

|x::xs ->

if a = x then

xs

else

x::(replace\_neg xs a)

in

let rec replace\_val l1 l2 a =

match l1 with

|[] -> l2

|x::xs ->

if x = a then

[]

else

(replace\_val xs (l2@[x]) a)

in

let reverse a =

match a with

|Plus(x) -> Minus(x)

|Minus(x) -> Plus(x)

in

let all\_valeurs\_forces l =

match l with

|[] -> ()

|x::xs ->

match x with

|Plus(y) ->

for i = 0 to q-1 do

begin

tab.(i) <- replace\_val tab.(i) [] x ;

tab.(i) <- replace\_neg tab.(i) (reverse x)

end

done

|Minus(y) ->

for i = 0 to q-1 do

begin

tab.(i) <- replace\_val tab.(i) [] (reverse x) ;

tab.(i) <- replace\_neg tab.(i) x

end

done

in

all\_valeurs\_forces valeurs\_forces;

for j = 0 to q-1 do

m:= List.length(tab.(j))+ !m

done;

!m