

```

theory Problem-4
  imports Main
begin

```

Let a and b be positive integers, and let A and B be finite sets of integers satisfying

- A and B are disjoint.
- if an integer i belongs to either A or B then either $i + a$ belongs to A or $i - b$ belongs to B .

Prove that $a|A| = b|B|$.

```

context
  fixes  $a\ b :: int$ 
  fixes  $A\ B :: int\ set$ 
  assumes  $pos: a > 0\ b > 0$ 
  assumes  $finite: finite\ A\ finite\ B$ 
  assumes  $disjoint: A \cap B = \{\}$ 
  assumes  $property: \forall i \in A \cup B. i + a \in A \vee i - b \in B$ 
begin

```

```

definition  $allows$  (infix  $allows\ 50$ ) where
   $x\ allows\ i \longleftrightarrow (x \in A \wedge i + a = x) \vee (x \in B \wedge i - b = x)$ 

```

```

lemma  $allows-right-unique$ :
  assumes  $x\ allows\ i$  and  $x\ allows\ j$ 
  shows  $i = j$ 
proof ( $rule\ ccontr$ )
  assume  $i \neq j$ 
  with  $assms$  have  $x \in A \wedge x \in B$  unfolding  $allows-def$  by  $auto$ 
  with  $disjoint$  show  $False$  by  $auto$ 
qed

```

```

definition  $allowers$  where
   $allowers\ i = \{x. x\ allows\ i\}$ 

```

```

lemma  $has-allowers: i \in A \cup B \implies allowers\ i \neq \{\}$ 
  unfolding  $allowers-def\ allows-def$  using  $property$  by  $auto$ 

```

```

lemma  $allowers-in-set: allowers\ i \subseteq A \cup B$ 
  unfolding  $allowers-def\ allows-def$  by  $auto$ 

```

```

lemma  $allowers-finite: finite\ (allowers\ i)$ 
  using  $allowers-in-set$  apply ( $rule\ finite-subset$ )
  using  $finite$  by  $auto$ 

```

```

lemma  $card-allowers: i \in A \cup B \implies card\ (allowers\ i) > 0$ 
  using  $has-allowers\ allowers-finite$  by  $fastforce$ 

```

```

lemma  $allowers-inj: inj-on\ allowers\ (A \cup B)$ 
proof ( $rule\ inj-onI$ )
  fix  $i\ j$ 
  assume  $i \in A \cup B\ j \in A \cup B$ 
  with  $has-allowers$  obtain  $x$  where  $x \in allowers\ i$  by  $auto$ 
  moreover assume  $allowers\ i = allowers\ j$ 
  ultimately have  $x\ allows\ i$  and  $x\ allows\ j$ 
  by ( $auto\ simp: allowers-def$ )
  thus  $i = j$  by ( $rule\ allows-right-unique$ )
qed

```

```

lemma  $one-allower$ :
  assumes  $i \in A \cup B$ 

```

```

shows card (allowers i) = 1
proof -
let ?C = allowers ` (A ∪ B)
have pairwise disjnt ?C
proof
fix P Q
assume P ∈ ?C and Q ∈ ?C and P ≠ Q
then obtain i and j
  where *: allowers i = P allowers j = Q
  and i ∈ A ∪ B j ∈ A ∪ B
  by auto
with ⟨P ≠ Q⟩ have i ≠ j by auto
{
  fix x
  assume x ∈ P and x ∈ Q
  with * have x allows i and x allows j
  unfolding allowers-def by auto
  with allows-right-unique have i = j by auto
  with ⟨i ≠ j⟩ have False..
}
thus disjnt P Q unfolding disjnt-def by auto
qed
moreover have P ∈ ?C ⟹ finite P for P
  using allowers-finite by auto
ultimately have sum-card: card (⋃ ?C) = sum card ?C
  by (intro card-Union-disjoint; auto)

have ⋃ ?C ⊆ A ∪ B
  apply (intro Union-least)
  using allowers-in-set by auto
hence card (⋃ ?C) ≤ card (A ∪ B)
  using finite by (intro card-mono; auto)
moreover have card ?C = card (A ∪ B)
  using allowers-inj by (intro card-image)
ultimately have sum-card-le: sum card ?C ≤ card ?C
  using sum-card by simp

show card (allowers i) = 1 when ¬ card (allowers i) > 1
  using card-allowers assms that by force
show ¬ card (allowers i) > 1
proof
assume card (allowers i) > 1
moreover have sum card ?C = card (allowers i) + sum card (?C - {allowers i})
  apply (intro sum.remove)
  using finite assms by auto
moreover have sum card (?C - {allowers i}) ≥ card (?C - {allowers i})
  apply (intro sum-bounded-below[where K=(1::nat), simplified])
  using card-allowers by fastforce
moreover have Suc (card (?C - {allowers i})) = card ?C
  apply (intro card.remove[symmetric])
  using finite assms by auto
ultimately have sum card ?C > card ?C
  by auto
with sum-card-le show False by simp
qed
qed

```

definition *allower* where
allower i = the-elem (allowers i)

lemma *the-elem-in-set*:
 assumes *is-singleton S*
 shows *the-elem S* ∈ *S*

using *assms* **by** (*metis is-singleton-the-elem singletonI*)

lemma *lower-allows*:

assumes $i \in A \cup B$

shows *lower* $i \in$ *allows* i **and** (*lower* i) *allows* i

proof –

from *assms* **have** *is-singleton* (*allows* i)

using *one-lower is-singleton-altdef* **by** *blast*

thus *lower* $i \in$ *allows* i

unfolding *lower-def* **by** (*intro the-elem-in-set*)

thus (*lower* i) *allows* i

unfolding *allows-def..*

qed

lemma *lower-in-set*:

assumes $i \in A \cup B$

shows *lower* $i \in A \cup B$

using *allows-in-set lower-allows assms* **by** *auto*

lemma *lower-bij*: *bij-betw* *lower* $(A \cup B)$ $(A \cup B)$

unfolding *bij-betw-def*

proof

show *inj-on* *lower* $(A \cup B)$

apply (*intro inj-onI*)

by (*metis lower-allows(2) allows-right-unique*)

hence *card* (*lower* ‘ $(A \cup B)$) = *card* $(A \cup B)$

by (*intro card-image*)

moreover **have** *lower* ‘ $(A \cup B) \subseteq A \cup B$

using *lower-in-set* **by** *auto*

ultimately **show** *lower* ‘ $(A \cup B) = A \cup B$

using *finite* **by** (*intro card-subset-eq; auto*)

qed

theorem $a * \text{int}(\text{card } A) = b * \text{int}(\text{card } B)$

proof –

let $?A = \{i \in A \cup B. \text{lower } i \in A\}$

let $?B = \{i \in A \cup B. \text{lower } i \in B\}$

have *: $?A \cup ?B = A \cup B$

using *lower-in-set* **by** *auto*

have *disjoint'*: $?A \cap ?B = \{\}$

using *disjoint* **by** *auto*

have *in-a*: $i \in ?A \implies \text{lower } i = i + a$ **for** i

using *lower-allows allows-def*

by (*metis (mono-tags, lifting) IntI disjoint empty-iff mem-Collect-eq*)

have *in-b*: $i \in ?B \implies \text{lower } i = i - b$ **for** i

using *lower-allows allows-def*

by (*metis (mono-tags, lifting) IntI disjoint empty-iff mem-Collect-eq*)

have *lower* ‘ $?A \subseteq A$ **and** *lower* ‘ $?B \subseteq B$

by *auto*

moreover **have** *lower* ‘ $(?A \cup ?B) = A \cup B$

using *lower-bij unfolding * bij-betw-def* **by** *auto*

ultimately **have** *lower* ‘ $?A = A$ **and** *lower* ‘ $?B = B$

using *disjoint* **by** *auto*

moreover **have** *inj-on* *lower* $?A$ **and** *inj-on* *lower* $?B$

using *bij-betw-imp-inj-on lower-bij inj-on-subset ** **by** *blast+*

ultimately **have** *cards*: *card* $?A = \text{card } A$ *card* $?B = \text{card } B$

using *card-image* **by** *fastforce+*

have $\sum (A \cup B) = \text{sum } \text{lower} (A \cup B)$

using *sum.reindex-bij-betw[OF lower-bij, of $\lambda x. x$]*..

also **have** ... = *sum* *lower* $(?A \cup ?B)$

```

    using * by simp
  also have ... = sum allower ?A + sum allower ?B
    using finite disjoint' by (intro sum.union-disjoint; auto)
  also have ... = ( $\sum i \in ?A. i + a$ ) + ( $\sum i \in ?B. i - b$ )
    using in-a in-b sum.cong
    by (metis (no-types, lifting))
  also have ... = ( $\sum ?A + a * \text{int} (\text{card } ?A)$ ) + ( $\sum ?B - b * \text{int} (\text{card } ?B)$ )
    by (simp add: sum.distrib sum-subtractf)
  also have ... = ( $\sum ?A + \sum ?B$ ) +  $a * \text{int} (\text{card } A) - b * \text{int} (\text{card } B)$ 
    unfolding cards
    by (simp add: ac-simps)
  also have ... =  $\sum (?A \cup ?B) + a * \text{int} (\text{card } A) - b * \text{int} (\text{card } B)$ 
  proof -
    have  $\sum (?A \cup ?B) = \sum ?A + \sum ?B$ 
      using finite disjoint' by (intro sum.union-disjoint; auto)
    thus ?thesis by simp
  qed
  also have ... =  $\sum (A \cup B) + a * \text{int} (\text{card } A) - b * \text{int} (\text{card } B)$ 
    unfolding *..
  finally show  $a * \text{int} (\text{card } A) = b * \text{int} (\text{card } B)$ 
    by simp
qed

end
end

```