ABSTRACT

Title of thesis: APERY SETS OF NUMERICAL SEMIGROUPS

Monica Grace Madero-Craven, Master of Arts, 2003

Thesis directed by: Professor Lawrence C. Washington
Department of Mathematics

A *numerical semigroup* is a subset, **S** of the non-negative integers, $\mathbb{Z}_+$ which contains zero, is closed under addition, and whose complement in $\mathbb{Z}_+$ is finite. We discuss the basic properties of numerical semigroups as well as associated structures such as *relative ideals*. Further, we examine several finite subsets of **S** including the *Apery Set* and two of its subsets. Relationships between these subsets of **S** will allow us to give an equivalent definition for **S** to be *symmetric* as well as a necessary condition for **S** to be *almost symmetric*.

APERY SETS OF NUMERICAL SEMIGROUPS

by

Monica Grace Madero-Craven

Thesis submitted to the faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Arts
2003

Advisory Committee:

Professor Lawrence C. Washington, Chair
Professor William W. Adams
Professor Niranjan Ramachandran

DEDICATION

To all of my children: Richard (RC), James, Ashley, David, Danielle, and Kimberly.

ACKNOWLEDGEMENTS

I wish to thank all the faculty and staff of the United States Air Force Academy, Department of Mathematical Science. I especially want to thank Col Daniel Litwhiler for his continual support and belief in my ability to succeed. I wish to thank Dr. Kurt Herzinger for introducing me to numerical semigroups. I also wish to express my sincere gratitude for his patience and guidance, not to mention the countless hours he has devoted to me during the past two years. Finally, I want to give a special thanks to my friends and family whose support was vital for my success: Chris and Kate Truman, Mark Johnston, Herb Schreiber, my parents, Michael and Monica Madero and my three youngest children, David, Danielle, and Kimberly.

TABLE OF CONTENTS

INTRODUCTION

In this thesis we will investigate various finite subsets of a numerical semigroup. A *numerical semigroup* is a subset **S** of the non-negative integers $\mathbb{Z}_+$ which contains zero, is closed under addition, and whose complement in $\mathbb{Z}_+$ is finite. The numerical semigroup **S** is denoted by its generators, that is, if $a_1,\ldots,a_k$ are the generators of **S**, then $\mathbf{S} = \langle a_1,\ldots,a_k \rangle$. **S** is the set of values created by linear combinations of the generators with non-negative coefficients.

In Chapter 1 we establish the standard definitions and notations related to numerical semigroups. These include the multiplicity, Frobenius number, and the minimal generating set for **S**. We will also briefly discuss structures associated to numerical semigroups called relative ideals.

In Chapter 2 we conduct an investigation of the Apery Set of **S** denoted by $Ap(\mathbf{S})$ and two of its subsets, $Ap'(\mathbf{S})$ and $Ap^*(\mathbf{S})$. We will demonstrate a known relationship between $\mathbf{S}'$ and $Ap'(\mathbf{S})$ but provide a proof that is somewhat different from the one provided in [5]. Next we will completely establish the relationship between $Ap^*(\mathbf{S})$ and $H(\mathbf{S})$. We will provide an equivalent definition of symmetric in terms of $Ap^*(\mathbf{S})$.

Finally, we discuss the notion of **S** being almost symmetric and prove a necessary condition for it in terms of $Ap'(\mathbf{S})$ and $Ap^*(\mathbf{S})$. We also provide an example that shows this condition is not sufficient.

The appendix of this thesis contains the code for a program used extensively in the research for this paper. It allows the user to quickly calculate all of the items defined in this paper. The program can be utilized in any DOS environment.

1. BASICS AND BACKGROUND

We begin by establishing the basic definitions and notation commonly associated with numerical semigroups. For more background on the topic of numerical semigroups the reader is encouraged to see [2], [5], [6], and [7].

(1.1) **Definitions/Notation:** Let $\mathbb{Z}_+$ denote the non-negative integers. A *numerical semigroup* **S** is a subset of $\mathbb{Z}_+$ such that

1) $0 \in \mathbf{S}$,
2) **S** is closed under addition,
3) there exists an $x \in \mathbb{Z}_+ \backslash \mathbf{S}$ such that, $y \in \mathbf{S}$ for all $y > x$.

The largest integer not contained in **S** is called the *Frobenius number* of **S** and is denoted by *g(**S**)*. The number of elements in **S** smaller than g(**S**) is denoted by *n(**S**)*. The smallest positive element of **S** is called the *multiplicity of **S*** and is denoted by *m(**S**)*.

(1.2) **Definition:** We say that a numerical semigroup **S** is *symmetric* provided the following statement is true for all $z \in \mathbb{Z}$:

$$z \in \mathbf{S} \Leftrightarrow g(\mathbf{S}) - z \notin \mathbf{S}.$$

(1.3) **Example:** Let **S** = {0, 5, 6, 7, 10, 11, 12, 13, 14,→}, (where → indicates all numbers greater than 14 are included in **S**.) Then **S** is a numerical semigroup with *g(S)* = 9, *n(S)* = 4, and *m(S)* = 5. Since $8 \notin \mathbf{S}$ and $g(\mathbf{S}) - 8 = 9 - 8 = 1 \notin \mathbf{S}$ we see **S** is not symmetric.

(1.4) **Example:** Let **S** = {0, 6, 8, 11, 12, 14, 16, 17, 18, 19, 20, 22, 23, 24, 25, 26, 27,→}. Then **S** is a numerical semigroup with *g(S)* = 21, *n(S)* = 11, and *m(S)* = 6. It's easy to check that **S** is symmetric since for every $z \notin \mathbf{S}, g(\mathbf{S}) - z \in \mathbf{S}$.

The following two facts are common in the literature on numerical semigroups. We present them here with proofs.

(1.5) **Fact:** $n(\mathbf{S}) \le \dfrac{g(\mathbf{S})+1}{2}$.

Proof: Case 1: $g(\mathbf{S})$ is odd. Partition the set $\{0,1,\ldots,g(\mathbf{S})\}$ as follows: $\{0,g(\mathbf{S})\}$, $\{1,g(\mathbf{S})-1\}$, …, $\{\frac{g(\mathbf{S})-1}{2},\frac{g(\mathbf{S})+1}{2}\}$. The partition is composed of $\frac{g(\mathbf{S})+1}{2}$ subsets. If $n(\mathbf{S}) > \dfrac{g(\mathbf{S})+1}{2}$, then by the Pigeon Hole Principle we know that at least one of the sets has two elements in common with **S**. Thus there exists $s_1, s_2 \in \mathbf{S}$ such that $s_1 + s_2 = g(\mathbf{S})$. Since **S** is closed under addition we conclude $g(\mathbf{S}) \in \mathbf{S}$ which is a contradiction.

Case 2: *g(S)* is even. In this case we want to partition the set $\{0,1,\ldots,g(\mathbf{S})\}$ as follows: $\{0,g(\mathbf{S})\},\{1,g(\mathbf{S})-1\},\ldots,\{\frac{g(\mathbf{S})-2}{2},\frac{g(\mathbf{S})+2}{2}\},\{\frac{g(\mathbf{S})}{2}\}$. The partition is composed of $\frac{g(\mathbf{S})+2}{2}$

subsets. If $n(\mathbf{S}) > \dfrac{g(\mathbf{S})+1}{2}$ then either every subset in the partition has one element in common with **S** or one of the sets has two elements in common with **S**. In either case there exists $s_1, s_2 \in \mathbf{S}$ such that $s_1 + s_2 = g(\mathbf{S})$. Again we have a contradiction.

In either case we conclude $n(\mathbf{S}) \leq \dfrac{g(\mathbf{S})+1}{2}$.

(1.6) **Fact:** A numerical semigroup **S** is *symmetric* if and only if *g(S)* is odd and $n(\mathbf{S}) = \dfrac{g(\mathbf{S})+1}{2}$.

Proof: For the forward implication, assume that *g(S)* is even or $n(\mathbf{S}) < \frac{g(\mathbf{S})+1}{2}$.

If *g(S)* is even then $\frac{g(\mathbf{S})}{2} \notin \mathbf{S}$ (since **S** is closed under addition) and $g(\mathbf{S}) - \frac{g(\mathbf{S})}{2} = \frac{g(\mathbf{S})}{2} \notin \mathbf{S}$.

So by definition **S** is not symmetric. If $n(\mathbf{S}) < \frac{g(\mathbf{S})+1}{2}$, then following the notation from (1.5), we see that one of the subsets in the partition of $\{0, 1, \ldots, g(\mathbf{S})\}$ has no elements in common with **S** (otherwise we have $g(\mathbf{S}) \in \mathbf{S}$). Thus there exists $z \in \mathbb{Z}$ such that $z \notin \mathbf{S}$ and $g(\mathbf{S}) - z \notin \mathbf{S}$. We conclude **S** is not symmetric.

For the reverse implication, assume *g(S)* is odd and $n(\mathbf{S}) = \frac{g(\mathbf{S})+1}{2}$. Again following the notation in (1.5), we have that each subset in the partition of $\{0, 1, \ldots, g(\mathbf{S})\}$ has exactly one element in common with **S**. Thus for every element of the set $\{0, 1, \ldots, g(\mathbf{S})\}$, we have $z \in \mathbf{S} \Leftrightarrow g(\mathbf{S}) - z \notin \mathbf{S}$. If $z < 0$ or $z > g(\mathbf{S})$, then it follows from our definitions that $z \in \mathbf{S} \Leftrightarrow g(\mathbf{S}) - z \notin \mathbf{S}$. We conclude **S** is symmetric.

(1.7) **Definition/Notation:** The *minimal generating set of **S*** is the unique smallest subset of **S** such that every element of **S** can be expressed as a linear combination of the elements in this subset with non-negative coefficients. We denote the size of the minimal generating set by $\mu(\mathbf{S})$.

If $\mu(\mathbf{S}) = k$ and the elements of the minimal generating set are $a_1, a_2, \ldots, a_k$ then the numerical semigroup is denoted by $\mathbf{S} = \langle a_1, a_2, \ldots, a_k \rangle = \{n_1 a_1 + \cdots + n_k a_k : n_1, \ldots, n_k \in \mathbb{Z}_+\}$, where $0 < a_1 < a_2 < \cdots < a_k$ and $a_m \notin \langle a_1, \ldots, a_{m-1} \rangle$.

(1.8) **Examples:** From (1.3), **S** = {0, 5, 6, 7, 10, 11, 12, 13, 14,→ }, can be expressed as $\mathbf{S} = \langle 5, 6, 7 \rangle = \{5k_1 + 6k_2 + 7k_3 \mid k_1, k_2, k_3 \in \mathbb{Z}_+\}$. Thus we have $\mu(\mathbf{S}) = 3$.

From (1.4), **S** = {0, 6, 8, 11, 12, 14, 16, 17, 18, 19, 20, 22, 23, 24, 25, 26, 27,→}, can be expressed as $\mathbf{S} = \langle 6, 8, 11 \rangle = \{6k_1 + 8k_2 + 11k_3 \mid k_1, k_2, k_3 \in \mathbb{Z}_+\}$ and again $\mu(\mathbf{S}) = 3$.

(1.9) **Fact:** Let **S** be a numerical semigroup with $\mu(\mathbf{S}) = 2$, that is $\mathbf{S} = \langle a_1, a_2 \rangle$. Then

(1) $g(\mathbf{S}) = a_1 a_2 - a_1 - a_2$ and

(2) **S** is symmetric.

Proof: The proofs of both these facts are common throughout the literature on numerical semigroups. In fact the proof of (1) is often found as a homework problem on linear Diophantine equations in textbooks on number theory (see [9], section 3.6, exercises 17,18). In Chapter 2 we will provide new proofs for both of these facts.

(1.10) **Definitions:** Given a semigroup **S**, we can derive a set from the elements not in **S** called the *holes of S*. We define the *holes of **S*** by

$$\mathrm{H}(\mathbf{S}) = \{z \in \mathbb{Z}_+ \mid z \notin \mathbf{S} \;\; and \;\; g(\mathbf{S}) - z \notin \mathbf{S}\}.$$

(In some papers $H(\mathbf{S})$ is referred to as the set of holes of the *second type*. See [3]).

From this definition we have an equivalent definition of what it means for **S** to be *symmetric*. The proof of the following fact is clear from the definitions.

(1.11) **Fact:** **S** is symmetric if only if $\mathrm{H}(\mathbf{S}) = \phi$.

(1.12) **Example:** If $\mathbf{S} = \langle 5, 6, 7 \rangle$, then *H(S)* = {1, 8}. If $\mathbf{S} = \langle 6, 8, 11 \rangle$, then **S** is symmetric, and we know from (1.11) that $\mathrm{H}(\mathbf{S}) = \phi$.

(1.13) **Definitions/Notation:** Let **S** be a numerical semigroup. A *relative ideal* is a nonempty subset **I** of $\mathbb{Z}$ such that **I** has a least element denoted by $m(\mathbf{I})$, and if $s \in \mathbf{S}$, and $i \in \mathbf{I}$, then $i + s \in \mathbf{I}$. There exists a largest element in $\mathbb{Z} \backslash \mathbf{I}$ called the *Frobenius number of I* and denoted by $g(\mathbf{I})$. A relative ideal **I** is usually denoted by its *minimal generating set* which is the unique smallest subset $T \subseteq \mathbf{I}$ such that every element of **I** can be expresses as $t + s$ where $t \in T$ and $s \in \mathbf{S}$. We denote the size of the minimal generating set of **I** by $\mu_{\mathbf{S}}(\mathbf{I})$. If $\mu_{\mathbf{S}}(\mathbf{I}) = n$ and the elements of the minimal generating set are $b_1, \ldots, b_n$ then the relative ideal is denoted by $\mathbf{I} = (b_1, \ldots, b_n) =$ $(b_1 + \mathbf{S}) \cup \cdots \cup (b_n + \mathbf{S})$ where $b_1 < \cdots < b_n$ and $b_m \notin (b_1, \ldots, b_{m-1})$.

(1.14) **Examples:** Let $\mathbf{S} = \langle 8,10,11,13\rangle$, $\mathbf{I} = (2,4)$, and $\mathbf{J} = (1,5)$. Then

$\mathbf{S} = \{0, 8, 10, 11, 13, 16, 18, 19, 20, 21, 22, 23, 24, 26, 27, 28, \to\}$, $g(\mathbf{S}) = 25$, $n(\mathbf{S}) = 13$

$\mathbf{I} = (2+\mathbf{S}) \cup (4+\mathbf{S}) = \{2, 4, 10, 12, 13, 14, 15, 17, 18, 20, 21, 22, 23, 24, 25, \to\}$

$g(\mathbf{I}) = 19$ and $m(\mathbf{I}) = 2$.

$\mathbf{J} = (1+\mathbf{S}) \cup (5+\mathbf{S}) = \{1, 5, 9, 11, 12, 13, 14, 15, 16, 17, 18, \to\}$ $g(\mathbf{J}) = 10$ and $m(\mathbf{J}) = 1$.

Note: It is clear from the definitions that $g(\mathbf{I}) \leq m(\mathbf{I}) + g(\mathbf{S})$.

(1.15) **Definitions:** If **I** and **J** are relative ideals of **S,** then we define $\mathbf{I}+\mathbf{J}$ and $\mathbf{I}-\mathbf{J}$ as $\mathbf{I}+\mathbf{J} = \{a+b \mid a \in \mathbf{I},\ b \in \mathbf{J}\}$ and $\mathbf{I}-\mathbf{J} = \{z \in \mathbb{Z} \mid z+\mathbf{J} \subseteq \mathbf{I}\}$. It is quick to check that both $\mathbf{I}+\mathbf{J}$ and $\mathbf{I}-\mathbf{J}$ are relative ideals of **S**. We call $\mathbf{I}-\mathbf{J}$ *the dual of* **J** *in* **I**. In the case when **J** = **S** we simply call this the *dual of* **I**.

(1.16) **Example:** As in (1.14) let $\mathbf{S} = \langle 8,10,11,13\rangle$, $\mathbf{I} = (2,4)$, and $\mathbf{J} = (1,5)$. Then

$\mathbf{I}+\mathbf{J} = \{3,5,7,9,11,13,14,15,16,17,18,19, \to\}$ = (3, 5, 7, 9).

$\mathbf{I}-\mathbf{J} = \{9,12,13,16,17,19,20,21,22,23,25,26, \to\}$ = (9, 12, 13, 16).

$\mathbf{S}-\mathbf{I} = \{6,9,14,16,17,18,19,20,22,24,25,26, \to\}$ = (6, 9, 18).

(1.17) **Definitions/Notation:** We define the *maximal ideal of* ***S*** to be **M** = **S**\{0}. Further we define $\mathbf{S}'$ = (**S**-**M**)**S**. The number of elements in $\mathbf{S}'$ is referred to as the *type of* ***S***.

(1.18) **Example:** From example (1.14), we can determine $\mathbf{M} = (8,10,11,13) = \{8,10,11,13,16,18,19,20,21,22,23,24,26,27,28,\rightarrow\}$. It is quick to confirm that **S** is symmetric by (1.16). Further we see $\mathbf{S}' = \{25\}$ and hence has type 1.

(1.19) **Facts:** (1) For any numerical semigroup **S** we have $g(\mathbf{S}) \in \mathbf{S}'$.

(2) The second largest element of $\mathbf{S}'$ is the largest element of H(**S**).

(3) **S** is symmetric if and only if $\mathbf{S}' = \{g(\mathbf{S})\}$; that is, if and only if **S** has type 1.

Proof: The proof of (1) is clear from the definition of $\mathbf{S}'$.

For (2), let h(**S**) denote the largest element of H(**S**). Let $s \in \mathbf{M}$. Then h(**S**) + s > h(**S**), so h(**S**) + s $\notin$ H(**S**). Suppose h(**S**) + s $\notin$ **S**. Then $g(\mathbf{S}) - (\mathrm{h}(\mathbf{S})) + s \in \mathbf{S}$ and hence $g(\mathbf{S}) - \mathrm{h}(\mathbf{S}) - s = t$ for some $t \in \mathbf{S}$. But $g(\mathbf{S}) - \mathrm{h}(\mathbf{S}) = s + t \in \mathbf{S}$ which is a contradiction. We must conclude that $\mathrm{h}(\mathbf{S}) \in \mathbf{S}'$ by definition.

Next suppose $z \in \mathbf{S}'$ and $z > \mathrm{h}(\mathbf{S})$. Consider $g(\mathbf{S}) - z$. Because $z > \mathrm{h}(\mathbf{S})$, we know $z \notin \mathrm{H}(\mathbf{S})$, and so we must have $g(\mathbf{S}) - z \in \mathbf{S}$. Thus $g(\mathbf{S}) = z + s$ for some $s \in \mathbf{S}$. If $s \in \mathbf{M}$, then $z + s \in \mathbf{S}$, and so $g(\mathbf{S}) \in \mathbf{S}$ which is a contradiction. We conclude that $s = 0$ and so $g(\mathbf{S}) = z$. Then there are no elements of $\mathbf{S}'$ strictly between h(**S**) and $g(\mathbf{S})$. The proof of (3) follows quickly from (1) and (2).

Note: A slightly different proof of (1.19(2)) can be found in [5].

Connections to Rings

Beyond being an interesting algebraic structure in their own right, numerical semigroups are often used as a tool to investigate problems in the area of commutative algebra. In particular, let **R** represent the power series ring $k[[t^{a_1},\ldots,t^{a_n}]]$, where k is a field and $0 < a_1 < \cdots < a_n$. Let υ represent the standard valuation mapping from the quotient field of **R** to $\mathbb{Z}$. Then $\upsilon(\mathbf{R})$ is the numerical semigroup $\mathbf{S} = \langle a_1,\ldots,a_k \rangle$, and many of the properties of **R** are reflected by the properties of **S**. For example:

(1) The embedding dimension of **R** = $\mu(\mathbf{S})$.

(2) If $\mathbf{I} = (t^{b_1},\ldots,t^{b_m})$ is a fractional ideal of **R**, then $\upsilon(\mathbf{I}) = (b_1,\ldots,b_m)$ is a relative ideal of **S**. Moreover, $\mu_{\mathbf{S}}(\upsilon(\mathbf{I})) = \mu_{\mathbf{R}}(\mathbf{I})$ and $\upsilon(\mathbf{I}^{-1}) = \mathbf{S} - \upsilon(\mathbf{I})$ [6].

(3) **R** is Gorenstein if and only if **S** is symmetric [7].

For more details on the connections between numerical semigroups and commutative algebra, please refer to [2], [3], [4], [6], and [7].

2. THE APERY SET AND ITS SUBSETS

(2.1) **Definitions:** We define a partial ordering $\leq_{\mathbf{S}}$ on a numerical semigroup **S** by $x \leq_{\mathbf{S}} y$ provided $y - x \in \mathbf{S}$ (see also [5]).

(2.2) **Example:** Let $\mathbf{S} = \langle 7,12,13 \rangle = \{0,7,12,13,14,19,20,21,24,25,26,27,28,31,\rightarrow\}$. Based on this partial ordering, $7 \leq_{\mathbf{S}} 19$ since $19 - 7 = 12 \in \mathbf{S}$ but $13 \not\leq_{\mathbf{S}} 19$ because $19 - 13 = 6 \notin \mathbf{S}$.

(2.3) **Note:** The elements of $\mathbf{S}\backslash\{0\}$ that are minimal under this partial ordering are exactly the elements of the minimal generating set for **S.**

(2.4) **Definitions/Notation:** Let $n \in \mathbf{S} \setminus \{0\}$. We define the *Apery Set with respect to n* to be $Ap(\mathbf{S}, n) = \{s \in \mathbf{S} : s - n \notin \mathbf{S}\}$. The Apery Set with respect to *m(S)* is typically denoted by $Ap(\mathbf{S})$. That is $Ap(\mathbf{S}) = \{s \in \mathbf{S} : s - m(\mathbf{S}) \notin \mathbf{S}\}$.

(2.5) **Note/Notation:** It follows from the definition that $Ap(\mathbf{S})$ contains exactly one element of **S** from each congruence class modulo *m(***S***)*. Specifically $Ap(\mathbf{S})$ consists of the smallest element of **S** which is congruent to i for $i = 0, 1, \ldots, m(\mathbf{S}) - 1$. We denote the element of $Ap(\mathbf{S})$ which is congruent to $i \bmod m(\mathbf{S})$ by $\omega(i)$. We denote the largest element of $Ap(\mathbf{S})$ by ω'. Further, it is important to note that with this definition we have $g(\mathbf{S}) + m(\mathbf{S}) = \omega'$.

Apery sets (named after Roger Apery, see [1]) appear often in the standard literature on numerical semigroups (see [5]) and are represented by a variety of different notations. For the development which follows, it seems most natural to adopt the notation established in [8].

(2.6) **Example:** Let $\mathbf{S}_1 = \langle 7, 12, 13 \rangle = \{0, 7, 12, 13, 14, 19, 20, 21, 24, 25, 26, 27, 28, 31, \rightarrow\}$. Then $Ap(\mathbf{S}_1) = \{0, 12, 13, 24, 25, 36, 37\}$, and we see $\omega(5) = 12$, $\omega(1) = 36$ and $\omega(2) = 37 = \omega'$.

The following four lemmas establish some of the basic properties of $Ap(\mathbf{S})$.

(2.7) **Lemma:** Every integer z has a unique representation in the form $z = \omega(i) + lm(\mathbf{S})$ for some i and $l \in \mathbb{Z}$. Moreover, $z \in \mathbf{S}$ if and only if $l \geq 0$.

Proof: Let z be some non-negative integer. Then $z \equiv i \bmod m(\mathbf{S})$, for some i.
Since $\omega(i)$ is the smallest element of **S** congruent to $i \bmod m(\mathbf{S})$, we know $z \in \mathbf{S}$ if and only if $z \geq \omega(i)$ which is true if and only if $z = \omega(i) + lm(\mathbf{S})$ for some $l \geq 0$.

(2.8) **Lemma:** $\omega(i)+\omega(j)=\omega(i+j)+lm(\mathbf{S})$ for some $l \geq 0$.

Proof: $\omega(i) \equiv i \bmod m(\mathbf{S})$ and $\omega(j) \equiv j \bmod m(\mathbf{S})$.

Thus $\omega(i)+\omega(j) \equiv i+j \bmod m(\mathbf{S})$. Also $\omega(i)+\omega(j) \in \mathbf{S}$ since $\mathbf{S}$ is closed under addition. The result follows from (2.7).

(2.9) **Lemma:** If $z_1, z_2 \in \mathbf{S}$ and $z_1+z_2 \in Ap(\mathbf{S})$, then $z_1, z_2 \in Ap(\mathbf{S})$.

Proof: (By Contrapositive): Let $z_1, z_2 \in \mathbf{S}$. Assume $z_1 \notin Ap(\mathbf{S})$ or $z_2 \notin Ap(\mathbf{S})$. Then $z_1 = \omega(i)+l_1 m(\mathbf{S})$ and $z_2 = \omega(j)+l_2 m(\mathbf{S})$ where $l_1 > 0$ or $l_2 > 0$ Thus $z_1 + z_2 =$ $\omega(i)+\omega(j)+(l_1+l_2)m(\mathbf{S}) = \omega(i+j)+km(\mathbf{S})+(l_1+l_2)m(\mathbf{S})$, where $k \geq 0$. Thus $z_1+z_2 = \omega(i+j)+(k+l_1+l_2)m(\mathbf{S})$, where $k+l_1+l_2>0$. Hence $z_1+z_2 \notin Ap(\mathbf{S})$. This completes the proof.

(2.10) **Example:** The converse of (2.9) is not always true. Consider example (2.6). $13, 25 \in Ap(\mathbf{S}_1)$ and $13, 25 \in \mathbf{S}_1$, but clearly, $13+25=38 \notin Ap(\mathbf{S}_1)$.

(2.11) **Lemma:** $\omega(i)-\omega(j)=\omega(i-j)+lm(\mathbf{S})$ where $l \leq 0$.

Proof: Consider $\omega(i-j)+\omega(j)$.

By (2.8), $\omega(i-j)+\omega(j)=\omega(i-j+j)+km(\mathbf{S}) = \omega(i)+km(\mathbf{S})$ for some $k \geq 0$. Then $\omega(i-j)=\omega(i)-\omega(j)+km(\mathbf{S})$. So $\omega(i-j)-km(\mathbf{S})=\omega(i)-\omega(j)$. Now let $l=-k$, hence $\omega(i-j)+lm(\mathbf{S})=\omega(i)-\omega(j)$.

(2.12) **Definitions:** There are two subsets of the Apery set which are of particular interest to this investigation:

$$Ap'(\mathbf{S}) = \{\omega \in Ap(\mathbf{S}) \mid \omega \text{ is maximal among the elements of } \mathrm{Ap}(\mathbf{S}) \text{ w.r.t. } \leq_s\} \text{ and}$$

$$Ap^*(\mathbf{S}) = \{\omega \in Ap(\mathbf{S}) \mid \omega' - \omega \notin \mathbf{S}\}.$$

(2.13) **Lemma:** $Ap'(\mathbf{S}) \subseteq Ap^*(\mathbf{S}) \cup \{\omega'\}$.

Proof: Clear from the definitions of $Ap'(\mathbf{S})$ and $Ap^*(\mathbf{S})$.

(2.14) **Example:** Let $\mathbf{S}_2 = \langle 8,11,12,15 \rangle = \{0,8,11,12,15,16,19,20,22,23,24,26,27,28,30,\rightarrow\}$.

Then $Ap(\mathbf{S}_2) = \{0,11,12,15,22,26,33,37\}$, $Ap'(\mathbf{S}_2) = \{12,33,37\}$, and $Ap^*(\mathbf{S}_2) = \{12,33\}$.

Notice $\omega' = 37$, and thus $Ap'(\mathbf{S}_2) \subseteq Ap^*(\mathbf{S}_2) \cup \{\omega'\}$.

The following lemma comes from [5] and reveals the bijective relationship between the sets $\mathbf{S}'$ and $Ap'(\mathbf{S})$. We offer a proof here that is slightly different than the one in [5].

(2.15) **Lemma:** $z \in \mathbf{S}'$ if and only if $z + m(\mathbf{S}) \in Ap'(\mathbf{S})$

Proof: Let $z \equiv i \mod m(\mathbf{S})$.

For the forward direction, assume $z \in \mathbf{S}'$. Then $z \notin \mathbf{S}$ but $z + m(\mathbf{S}) \in \mathbf{S}$. Thus $z + m(\mathbf{S})$ is the smallest element of $\mathbf{S}$ congruent to $i \mod m(\mathbf{S})$. By definition we have $z + m(\mathbf{S}) = \omega(\mathrm{i}) \in \mathrm{Ap}(\mathbf{S})$. Now suppose $\omega(j) - \omega(i) \in \mathbf{S}$ for some $j \neq i$. Then by (2.9) and (2.11) $\omega(j) - \omega(i) = \omega(j-i)$. Thus $\omega(j-i) = \omega(j) - z - m(\mathbf{S})$ so $z + \omega(j-i) = \omega(j) - m(\mathbf{S}) \notin \mathbf{S}$ by (2.7). This is a contradiction since $z \in \mathbf{S}'$ and $\omega(j-i) \in \mathbf{S} \setminus \{0\}$.

We conclude $\omega(j)-\omega(i)\notin \mathbf{S}$ for all $j\neq i$, and hence $\omega(i)$ is maximal in $Ap(\mathbf{S})$ with respect to $\leq_{\mathbf{S}}$. Therefore $\omega(i)\in Ap'(\mathbf{S})$.

For the reverse direction, assume $z+m(\mathbf{S})\in Ap'(\mathbf{S})$. Then $z+m(\mathbf{S})=\omega(i)$ and $\omega(i)$ is maximal in $Ap(\mathbf{S})$ with respect to $\leq_{\mathbf{S}}$. So by (2.7) we know $z=\omega(i)-m(\mathbf{S})\notin \mathbf{S}$. Now let $s\in \mathbf{S}\setminus\{0\}$, say $s=\omega(j)+lm(\mathbf{S})$ where $j\not\equiv 0 \bmod m(\mathbf{S})$ or $l>0$. Then $z+s=\omega(i)-m(\mathbf{S})+\omega(j)+lm(\mathbf{S})$. Note that $\omega(i+j)-\omega(i)\notin Ap(\mathbf{S})$ because $\omega(i)\in Ap'(\mathbf{S})$. Therefore $\omega(i)+\omega(j)=\ \omega(i+j)+km(\mathbf{S})$ where $k>0$. Thus, $z+s=\omega(i+j)+(k+l-1)m(\mathbf{S})$ where $k+l-1\geq 0$. So we have $z+s\in \mathbf{S}$ by (2.7). Since s was an arbitrary element of $\mathbf{S}\setminus\{0\}$, we conclude $z\in \mathbf{S}'$, by definition.

(2.16) **Example:** Using (2.15) we can determine for example (2.6) $\mathbf{S}_1'=\{29,30\}$ and for example (2.14) $\mathbf{S}_2'=\{4,25,29\}$.

(2.17) **Corollary:** $\mathbf{S}$ is symmetric if and only if $Ap'(\mathbf{S})=\{\omega'\}$.

Proof: The statement follows immediately from (2.15), (2.5), and (1.19(3)).

We now begin an examination of the properties of $Ap^*(\mathbf{S})$.

(2.18) **Proposition:** Assume $\omega(i),\omega(j)\in Ap(\mathbf{S})$ with the property that $\omega(i)+\omega(j)\in Ap(S)$. If $\omega(i)\in Ap^*(\mathbf{S})$ or $\omega(j)\in Ap^*(\mathbf{S})$, then $\omega(i)+\omega(j)\in Ap^*(\mathbf{S})$.

Proof: Suppose $\omega(i)\in Ap^*(\mathbf{S})$. Then $\omega'-\omega(i)\notin \mathbf{S}$. Let $\omega'\equiv k \bmod m(\mathbf{S})$, that is,

$\omega' = \omega(k)$. Then $\omega(k) - \omega(i) = \omega(k-i) - lm(\mathbf{S})$ where $l > 0$ by (2.7) and (2.11). Now consider $\omega' - (\omega(i) + \omega(j))$ which equals $\omega(k) - \omega(i) - \omega(j) = \omega(k-i) - \omega(j) - lm(\mathbf{S})$.

<u>Case 1</u>: $\omega(k-i) - \omega(j) \in \mathbf{S}$

In this case, $\omega(k-i) - \omega(j) = \omega(k-i-j)$ by (2.7) and (2.11). Thus

$\omega(k) - \omega(i) - \omega(j) = \omega(k-i-j) - lm(\mathbf{S}) \notin \mathbf{S}$. We conclude $\omega(k) - (\omega(i) + \omega(j)) \notin \mathbf{S}$.

<u>Case 2</u>: $\omega(k-i) - \omega(j) \notin \mathbf{S}$

In this case $\omega(k-i) - \omega(j) = \omega(k-i-j) - tm(S)$, where $t > 0$ by (2.7). Thus

$\omega(k) - \omega(i) - \omega(j) = \omega(k-i) - \omega(j) - lm(S) = \omega(k-i-j) - tm(S) - lm(S)$

$= \omega(k-i-j) - (t+l)m(S)$, where $t + l > 0$. Hence $\omega(k) - (\omega(i) - \omega(j) \notin \mathbf{S}$ by (2.7).

In both cases we have $\omega' - (\omega(i) - \omega(j)) \notin \mathbf{S}$. By definition of $Ap^*(\mathbf{S})$ we conclude that $\omega(i) + \omega(j) \in Ap^*(\mathbf{S})$.

(2.19) **Note/Example:** If $\omega(i), \omega(j) \in Ap(S) \setminus Ap^*(S)$, then $\omega(i) + \omega(j)$ may or may not be in $Ap^*(S)$. From example (2.6), $Ap(\mathbf{S}_1) = \{0,12,13,24,25,36,37\}$ and $Ap^*(\mathbf{S}_1) = \{36\}$. So we have $12,13,24 \in Ap(\mathbf{S}_1) \setminus Ap^*(\mathbf{S}_1)$ and $12 + 24 = 36 \in Ap^*(\mathbf{S}_1)$ but $12 + 13 = 25 \notin Ap^*(\mathbf{S}_1)$.

(2.20) **Lemma:** If $\omega(i) \in Ap^*(\mathbf{S})$, then $\omega(i) - m(\mathbf{S}) \in H(\mathbf{S})$.

Proof: Let $\omega(i) \in Ap^*(\mathbf{S})$ then by (2.7) $\omega(i) - m(\mathbf{S}) \notin \mathbf{S}$. Also $g(\mathbf{S}) - [\omega(i) - m(\mathbf{S})] = g(\mathbf{S}) - \omega(i) + m(\mathbf{S}) = \omega' - \omega(i) \notin \mathbf{S}$. Hence by definition $\omega(i) - m(\mathbf{S}) \in H(\mathbf{S})$.

(2.21) **Lemma:** If $z \in H(\mathbf{S})$, then $z = \omega(i) - lm(\mathbf{S})$ where $l > 0$ and $\omega(i) \in Ap^*(\mathbf{S})$.

Proof: Let $z \in H(\mathbf{S})$, then $z \notin \mathbf{S}$. By (2.7), $z = \omega(i) - lm(\mathbf{S})$ for some i and some $l > 0$. We need only to show $\omega(i) \in Ap^*(\mathbf{S})$.

Now consider $\omega' - \omega(i) = g(\mathbf{S}) + m(\mathbf{S}) - [z + lm(\mathbf{S})] = g(\mathbf{S}) + m(\mathbf{S}) - z - lm(\mathbf{S})$. So $\omega' - \omega(i) = g(\mathbf{S}) - z + (1-l)m(\mathbf{S})$. But $g(\mathbf{S}) - z \notin \mathbf{S}$, so $g(\mathbf{S}) - z = \omega(j) - km(\mathbf{S})$ for some j and some $k > 0$ by (2.7). Thus $\omega' - \omega(i) = \omega(j) + (1-l-k)m(\mathbf{S})$ where $1-l-k < 0$. Thus $\omega' - \omega(i) \notin \mathbf{S}$ by (2.7), and we conclude $\omega(i) \in Ap^*(\mathbf{S})$.

The previous two lemmas reveal the following fact about symmetry.

(2.22) **Fact:** **S** is symmetric if and only if $Ap^*(\mathbf{S}) = \phi$.

Proof: By (2.20) and (2.21) we have $Ap^*(\mathbf{S}) = \phi$ if and only if $H(\mathbf{S}) = \phi$ which is true if and only if **S** is symmetric by (1.11).

As promised in Chapter 1, we now provide a proof of (1.9) from the standpoint of Apery Sets.

Let $\mathbf{S} = \langle a_1, a_2 \rangle$. Then for all $s \in \mathbf{S}$ we know $s = k_1 a_1 + k_2 a_2$ where $k_1, k_2 \geq 0$. Notice that if $k_1 \geq 1$ then $s \notin Ap(\mathbf{S})$ because $s - a_1 = (k_1 - 1)a_1 + k_2 a_2 \in \mathbf{S}$, (recall that $a_1 = m(\mathbf{S})$.) Thus $s \in Ap(\mathbf{S})$ if and only if $k_1 = 0$ and $0 \leq k_2 \leq a_1 - 1$. We then conclude that $Ap(\mathbf{S}) = \{0, a_2, 2a_2, ..., (a_1 - 1)a_2\}$. Now by (2.5), we see $g(\mathbf{S}) = \omega' - m(\mathbf{S}) = (a_1 - 1)a_2 - a_1 = a_1 a_2 - a_1 - a_2$. Next notice that if $0 \leq j \leq a_1 - 1$, then $\omega' - ja_2 =$

$(a_1-1)a_2 - ja_2 = (a_1-1-j)a_2 \in \mathbf{S}$. Therefore $ja_2 \notin Ap^*(\mathbf{S})$. Hence $Ap^*(\mathbf{S}) = \phi$. Thus **S** is symmetric by (2.22).

(2.23) **Lemma:** If $\omega(i) - lm(\mathbf{S}) \notin H(\mathbf{S})$, then $\omega(i) - (l+1)m(\mathbf{S}) \notin H(\mathbf{S})$ where $l > 0$.

Proof: (By Contrapositive) Let $l > 0$ and assume $\omega(i) - (l+1)m(\mathbf{S}) \in H(\mathbf{S})$. Then $g(\mathbf{S}) - \omega(i) + (l+1)m(\mathbf{S}) \notin \mathbf{S}$. So $g(\mathbf{S}) - \omega(i) + (l+1)m(\mathbf{S}) = g(\mathbf{S}) - \omega(i) + lm(\mathbf{S}) + m(\mathbf{S})$ $= g(\mathbf{S}) + m(\mathbf{S}) - [\omega(i) - lm(\mathbf{S})] \notin \mathbf{S}$. Thus $g(\mathbf{S}) - [\omega(i) - lm(\mathbf{S})] \notin \mathbf{S}$. So by definition $\omega(i) - lm(\mathbf{S}) \in H(\mathbf{S})$. This completes the proof.

(2.24) **Definition:** We define $H(\mathbf{S}, i) = \{z \in H(\mathbf{S}) \mid z \equiv i \bmod m(\mathbf{S})\}$.

(2.25) **Example:** From example (2.14), we have $H(\mathbf{S}_2) = \{4, 25\}$, so $H(\mathbf{S}_2, 4) = \{4\}$ and $H(\mathbf{S}_2, 1) = \{25\}$. If we look at example (2.6), we have $H(\mathbf{S}_1) = H(\mathbf{S}_1, 1) = \{1, 8, 15, 22, 29\}$.

The following two theorems completely establish the relationship between $Ap^*(\mathbf{S})$ and $H(\mathbf{S})$.

(2.26) **Theorem:** Let $i + j \equiv g(\mathbf{S}) \bmod m(\mathbf{S})$. Then $|H(\mathbf{S}, i)| = |H(\mathbf{S}, j)|$. Further, if $|H(\mathbf{S}, i)| = |H(\mathbf{S}, j)| = k$, then $H(\mathbf{S}, i) = \{\omega(i) - m(\mathbf{S}), \omega(i) - 2m(\mathbf{S}), \ldots, \omega(i) - km(\mathbf{S})\}$ and $H(\mathbf{S}, j) = \{\omega(j) - m(\mathbf{S}),\ \omega(j) - 2m(\mathbf{S}), \ldots, \omega(j) - km(\mathbf{S})\}$.

Proof: Let $x \in H(\mathbf{S}, i)$ then by definition we know $g(\mathbf{S}) - x \in H(\mathbf{S})$ and

$g(\mathbf{S}) - x \equiv j \bmod m(\mathbf{S})$. Thus $g(\mathbf{S}) - x \in H(\mathbf{S}, j)$. Similarly, if $y \in H(\mathbf{S}, j)$, then $g(\mathbf{S}) - y \in H(\mathbf{S}, i)$. We conclude $|H(\mathbf{S}, i)| = |H(\mathbf{S}, j)|$.

Assume $|H(\mathbf{S}, i)| = k$. If $k = 0$, then there is nothing to prove. Let $k > 0$. By the definition of $H(\mathbf{S}, i)$ and (2.7) we know every element of $H(\mathbf{S}, i)$ must be of the form $\omega(i) - lm(\mathbf{S})$ where $l > 0$. Let $1 \le t \le k$. If $\omega(i) - tm(\mathbf{S}) \notin H(\mathbf{S})$, then by (2.23) we know $\omega(i) - vm(\mathbf{S}) \notin H(\mathbf{S})$ for $v \ge t$. Hence $|H(\mathbf{S}, i)| < t \le k$, which is a contradiction. Thus $\{\omega(i) - m(\mathbf{S}), \omega(i) - 2m(\mathbf{S}), ..., \omega(i) - km(\mathbf{S})\} \subseteq H(\mathbf{S}, i)$. Since $|H(\mathbf{S}, i)| = k$, we have our conclusion. The proof for $|H(\mathbf{S}, j)|$ is similar.

(2.27) **Theorem:** If $\omega(i) + \omega(j) = \omega' + km(\mathbf{S})$, then $|H(\mathbf{S}, i)| = |H(\mathbf{S}, j)| = k$.

Proof: Clearly $|H(\mathbf{S}, i)| = |H(\mathbf{S}, j)|$, by (2.26)

Assume $\omega(i) + \omega(j) = \omega' + km(\mathbf{S})$. If $k = 0$, then $\omega(i) + \omega(j) = \omega'$. Thus $\omega(i) \notin Ap^*(\mathbf{S})$ and $\omega(j) \notin Ap^*(\mathbf{S})$. So by (2.21), we may conclude $H(\mathbf{S}, i) = H(\mathbf{S}, j) = \phi$. Now assume $k \ge 1$. We will show $\omega(i) - km(\mathbf{S}) \in H(\mathbf{S})$ and $\omega(i) - (k+1)m(\mathbf{S}) \notin H(\mathbf{S})$.

We know $\omega(i) - km(\mathbf{S}) \notin \mathbf{S}$ by (2.7). Now consider $g(\mathbf{S}) - (\omega(i) - km(\mathbf{S})) =$ $g(\mathbf{S}) - \omega(i) + km(\mathbf{S}) = g(\mathbf{S}) + m(\mathbf{S}) - \omega(i) + (k-1)m(\mathbf{S}) = \omega' - \omega(i) + (k-1)m(\mathbf{S}) =$ $\omega(j) - km(\mathbf{S}) + (k-1)m(\mathbf{S}) = \omega(j) - m(\mathbf{S}) \notin \mathbf{S}$ by (2.7). Hence $\omega(i) - km(\mathbf{S}) \in H(\mathbf{S})$.

To show that $\omega(i) - (k+1)m(\mathbf{S}) \notin H(\mathbf{S})$ we consider $g(\mathbf{S}) - (\omega(i) - (k+1)m(\mathbf{S})) =$ $g(\mathbf{S}) - \omega(i) + (k+1)m(\mathbf{S}) = g(\mathbf{S}) + m(\mathbf{S}) - \omega(i) + km(\mathbf{S}) = \omega' - \omega(i) + km(\mathbf{S}) = \omega(j) \in \mathbf{S}$.

By definition, $\omega(i)-(k+1)m(\mathbf{S})\notin H(\mathbf{S})$. Now by (2.23) we know $\omega(i)-tm(\mathbf{S})\notin H(\mathbf{S})$ for $t\geq k+1$. Thus $|H(\mathbf{S},i)|=|H(\mathbf{S},j)|=k$.

(2.28) **Example:** From example (2.14), we have $Ap^*(\mathbf{S}_2)=\{12,33\}$ where $\omega(4)=12$ and $\omega(1)=33$. Since $\omega'=\omega(5)=37$, we consider $\omega(1)+\omega(4)=\omega'+m(\mathbf{S}_2)$. This tells us that $|H(\mathbf{S}_2,1)|=|H(\mathbf{S}_2,4)|=1$, which agrees with what we determined in (2.25). Now we look at example (2.6) where $Ap^*(\mathbf{S}_1)=\{36\}$, $\omega(1)=36$, and $\omega'=\omega(2)=37$. So we have $\omega(1)+\omega(1)=\omega'+5m(\mathbf{S}_1)$. This tells us that $|H(\mathbf{S}_1,1)|=5$, which again agrees with what we stated in (2.25).

When a numerical semigroup **S** is not symmetric, it is natural to inquire as to "how far it is from being symmetric." Throughout the study of numerical semigroups various measures of symmetry have been devised. Those semigroups which are considered "close" to being symmetric are often given special names. For example, if $g(\mathbf{S})$ is even and $H(\mathbf{S})=\left\{\frac{g(\mathbf{S})}{2}\right\}$, then **S** is said to be *psuedosymmetri*c [2]. The concept of *almost symmetric* was introduced in [3]. We give the definition here as well.

(2.29) **Definition:** We say **S** is *almost symmetric* provided $\mathbf{S}'=H(\mathbf{S})\cup\{g(\mathbf{S})\}$.

(2.30) **Example:** Using this definition we can quickly determine if our two examples are almost symmetric. Since $\mathbf{S}_1'=\{29,30\}$ and $H(\mathbf{S}_1)=\{1,8,15,22,29\}$ clearly $\mathbf{S}_1$ is not almost symmetric. Next, we see that $\mathbf{S}_2' = \{4, 25, 29\}$, $H(\mathbf{S}_2) = \{4, 25\}$, and $g(\mathbf{S}_2) = 29$. Thus $\mathbf{S}_2$ is almost symmetric since $\mathbf{S}_2' = H(\mathbf{S}_2) \cup \{g(\mathbf{S}_2)\}$.

The following theorem gives a necessary condition, in terms of Apery Sets, for **S** to be almost symmetric. However, the example which follows the theorem shows that this condition is not sufficient.

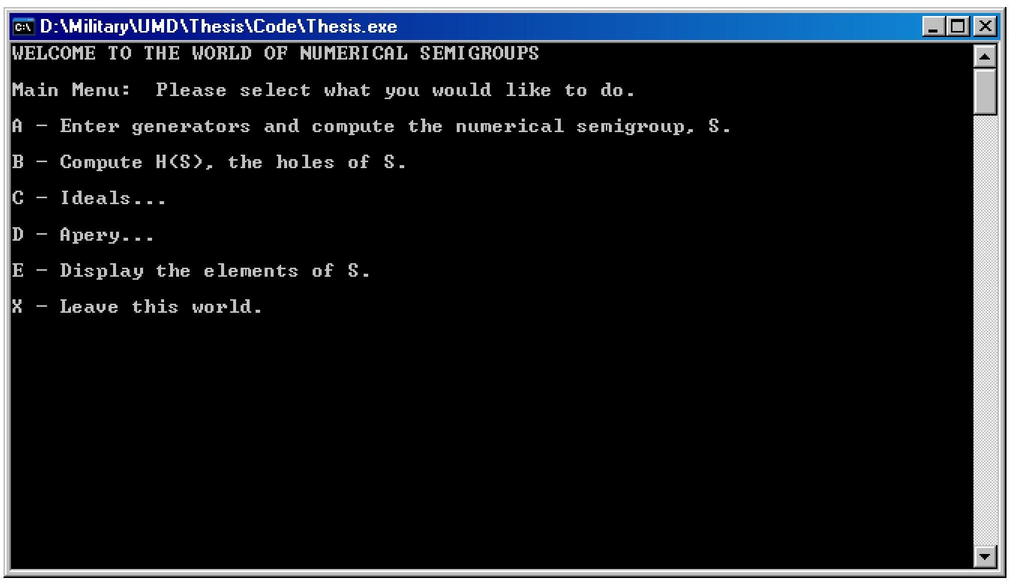
(2.31) **Theorem:** If **S** is almost symmetric, then $Ap'(\mathbf{S}) = Ap^*(\mathbf{S}) \cup \{\omega'\}$.

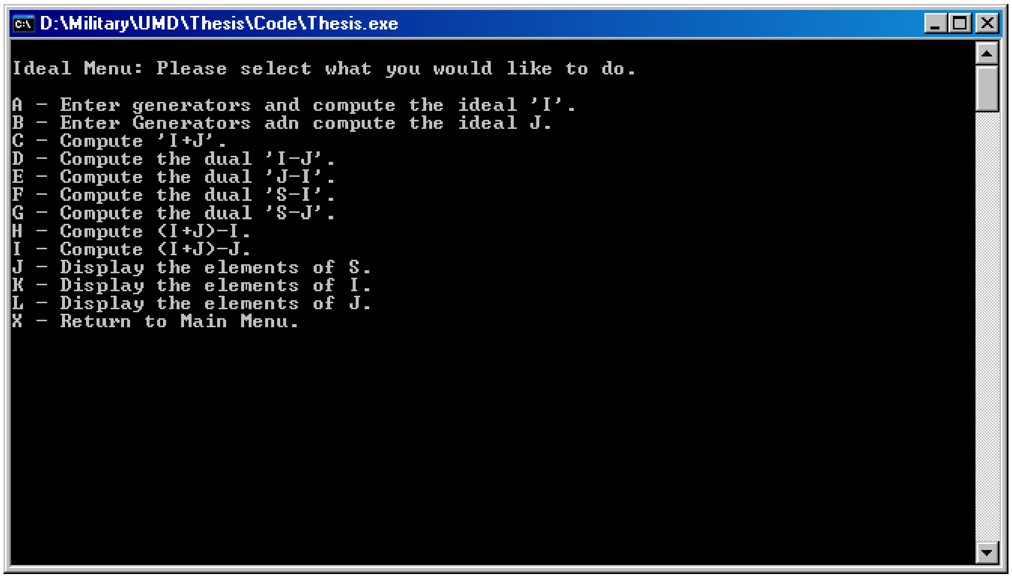
Proof: (By Contrapositive): Suppose $Ap'(\mathbf{S}) \neq Ap^*(\mathbf{S}) \cup \{\omega'\}$. By (2.13) there exists some $\omega(i) \in Ap^*(\mathbf{S}) \setminus Ap'(\mathbf{S})$. So $\omega(i) - m(\mathbf{S}) \in H(\mathbf{S})$ by (2.20). But $\omega(i) - m(\mathbf{S}) \notin \mathbf{S}'$ because $\omega(i) \notin Ap'(\mathbf{S})$ by (2.15). Therefore $\mathbf{S}' \neq H(\mathbf{S}) \cup \{g(\mathbf{S})\}$, whence **S** is not almost symmetric by definition. This completes the proof.

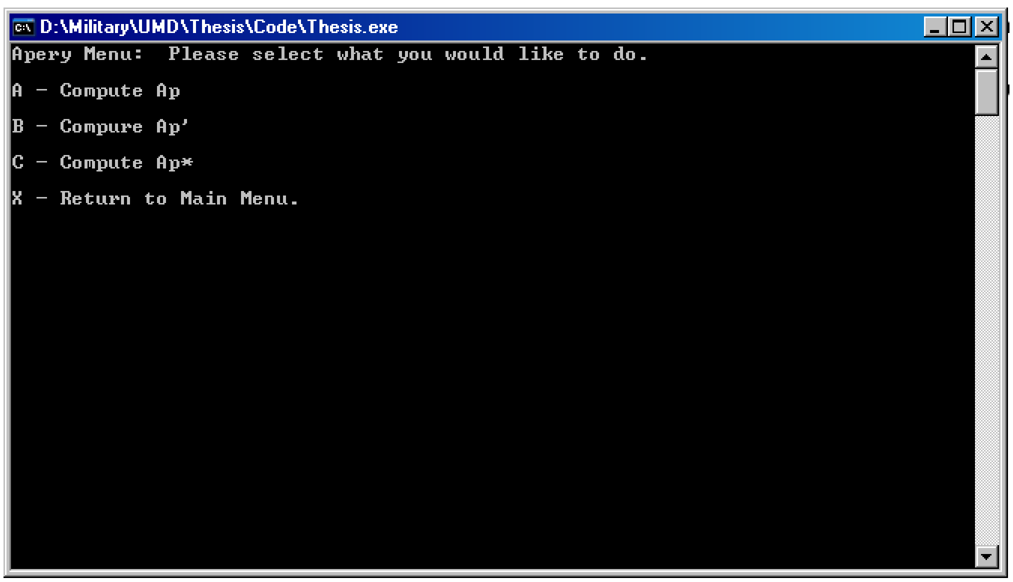
(2.32) **Example:** If we use this theorem to check example (2.14), we see that since $\mathbf{S}_2$ is almost symmetric $Ap'(\mathbf{S}_2) = Ap^*(\mathbf{S}_2) \cup \{\omega'\}$. If we look at $\mathbf{S}_1$ (example 2.6) we see that $Ap'(\mathbf{S}_1) = Ap^*(\mathbf{S}_1) \cup \{\omega'\}$, but we know from (2.30) that $\mathbf{S}_1$ is not almost symmetric. So the converse is not always true, that is, $Ap'(\mathbf{S}) = Ap^*(\mathbf{S}) \cup \{\omega'\}$ does not imply almost symmetric.

APPENDIX

This appendix contains the code for a program used extensively in the research for this paper. It allows the user to quickly calculate all of the items defined in this paper. The program can be utilized in any DOS environment. The menus available to the user are provided below.




D:\Military\UMD\Thesis\Code\Thesis.exe
Ideal Menu: Please select what you would like to do.
A - Enter generators and compute the ideal 'I'.
B - Enter Generators adn compute the ideal J.
C - Compute 'I+J'.
D - Compute the dual 'I-J'.
E - Compute the dual 'J-I'.
F - Compute the dual 'S-I'.
G - Compute the dual 'S-J'.
H - Compute (I+J)-I.
I - Compute (I+J)-J.
J - Display the elements of S.
K - Display the elements of I.
L - Display the elements of J.
X - Return to Main Menu.


D:\Military\UMD\Thesis\Code\Thesis.exe
Apery Menu: Please select what you would like to do.
A - Compute Ap
B - Compure Ap'
C - Compute Ap*
X - Return to Main Menu.

```
//=============================================================
// Numerical Semigroups
// By Capt Monica Madero-Craven
//=============================================================

#include <iostream.h>
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>

//Variables
  int i;           //index for loops
  int flag = 12;   // number of items tracked in flag array
  int flags[12]; //array to track computed items
  int max_s, max_i,max_j, max_ij;   //maximum size for semigroup and ideals
  int max_dij, min_dij, max_dji, min_dji;      //max and min I-J and J-I
  int max_dsi, min_dsi, max_dsj, min_dsj;      //max and min S-I and S-J
  int max_dij_i, min_dij_i, max_dij_j, min_dij_j;  //max/min  (I+J)-I, (I+J)-I
  int max_ap;                     //maximum size for Apery
  int count_s, count_i, count_j;   //number of generators
  int count_ij;   //number of generators in I+J
  int generators_s[100];          //array for generators of S
  int generators_i[10], generators_j[10];  //arrays for generators of ideals
  int generators_ij[50];  //array for the generators of I+J
  int semigroup[1000];   //array for creating semigroup
  int ideal_i[100], ideal_j[100];  //array for creating ideals I and J
  int ideal_ij[100];  //array for creating I+J
  int dual_ij[100], dual_ji[100];   //array for creating I-J, J-I
  int dual_sj[100], dual_si[100];   //array for creating S-J, S-I
  int dual_ij_i[100], dual_ij_j[100];   //array for creating (I+J)-I, (I+J)-J
  int holes[1000];  //array for the holes of S
  int g_s, g_i, g_j, g_ij;    //Frobenius number
  int n_s, n_i, n_j, n_ij;    //number of elements in the set
  int apery[500], apery_prime[100], apery_star[100];
```

```
//Functions
  void initialize_array(int array[], int count);
  int many_generators(void);       //fcn to determine how many generators
  void get_generators(int gen[], int count); //fcn to get generators
  void include_gen(int gen[], int count, int array[]);
  void create_s(int gen[], int count, int group[], int maximum);
  void create_ideal(int gen[], int count, int group[], int g_s, int ideal[]);
  int find_frobenius(int group[], int g); //Find Frobenius
  int count_elements(int array[], int count);  //Count elements
  void print_array(int array[], int count);   //Print items  = 1
  void print_other(int g, int n);    //Print the Frobenius number and n
  char enter_s_error(void);     //user must enter S first
  char ideal_error(), apery_error();     //user needs another option first
  void add_ideals(int sum_ideal[], int gen_1[], int count_1, int gen_2[],
                int count_2);   //adds any two ideals
  void create_dual(int array_1[], int g_1, int gen_2[], int count_2, int dual[],
                 int minimum, int maximum);  //create dual of array1 - array2
  void create_apery(int gen[], int group[], int g_s, int apery[], int count);
  void create_prime(int group[], int apery[], int max_ap, int prime[]);
  void create_star(int group[], int apery[], int maximum, int star[]);
  void create_holes(int h[], int s[], int gs);

//Menu Variables and Functions
  char main_choice, ideal_choice;      //letter selected from the menu
  char apery_choice;   //letter selected form the menu
  char main_choice_menu(void);     //function to print main menu
  char ideal_choice_menu(void);    //function to print ideal menu
  char apery_choice_menu(void);    //function to print apery menu


int main()
{
//Welcome and choice menu.

 cout <<"WELCOME TO THE WORLD OF NUMERICAL SEMIGROUPS\n\n";
 main_choice = toupper(main_choice_menu());

 while (main_choice != 'X')
 {

//*******************    Enter Generators and Compute S   *******************

  if (main_choice == 'A')
  {
   initialize_array(flags,flag);
   flags[0]= 1;
```

```
    count_s = many_generators();
    get_generators(generators_s,count_s);
    max_s = (generators_s[0]-1)*(generators_s[1]-1)+1;
    initialize_array(semigroup,max_s);
    include_gen(generators_s,count_s,semigroup);
    create_s(generators_s,count_s,semigroup,max_s);
    g_s = find_frobenius(semigroup, max_s);
    n_s = count_elements(semigroup, g_s);
    system ("cls");
    cout << "\n\nS = ";
    print_array(semigroup, g_s);
    print_other(g_s, n_s);
    system ("Pause");
    system ("cls");
    main_choice = toupper(main_choice_menu());
    continue;
   }   //End of Main Choice A

//*******************       Compute Holes of S, H(S)       *******************

   if (main_choice == 'B')
   {
    if (flags[0] != 1)
    {
     cout << "\n\n";
     main_choice = toupper(enter_s_error());
     continue;
    }
    flags[1] = 1;
    initialize_array(holes,g_s+1);
    create_holes(holes,semigroup,g_s);
    system ("cls");
    cout << "\n\nH(S)= ";
    print_array(holes,g_s);
    system ("Pause");
    system ("cls");
    flags[1] = 0;
    main_choice = toupper(main_choice_menu());
    continue;
   }  //End of Main choice B

//*******************         Go to the Ideal Menu        *******************

   if (main_choice == 'C')
   {
    if (flags[0] != 1)        //verify S is alreay computed
```

```
  {
   main_choice = toupper(enter_s_error());
   continue;
  }
  ideal_choice = toupper(ideal_choice_menu());   //display menu
  while (ideal_choice != 'X')        //while != return to main
  {

/*******************   Enter Generators and Compute I   *******************/

   if (ideal_choice == 'A')
   {
    flags[2]= 1;
    flags[4] = flags[5] = flags[6] = flags[7] = 0;
    count_i = many_generators();
    get_generators(generators_i,count_i);
    max_i = g_s + generators_i[0]+1;
    initialize_array(ideal_i,max_i);
    create_ideal(generators_i,count_i,semigroup,g_s,ideal_i);
    g_i = find_frobenius(ideal_i, max_i);
    n_i = count_elements(ideal_i, g_i);
    system ("cls");
    cout << "\n\nI = ";
    print_array(ideal_i, g_i);
    print_other(g_i, n_i);
    system ("Pause");
    ideal_choice = toupper(ideal_choice_menu());
    continue;
   }  //End of Ideal Choice A

//*******************   Enter Generators and Compute J   *******************

   if (ideal_choice == 'B')        //enter generators and compute J
   {
    flags[3]= 1;
    flags[4] = flags[5] = flags[6] = flags[8] = 0;
    count_j = many_generators();
    get_generators(generators_j,count_j);
    max_j = g_s + generators_j[0]+1;
    initialize_array(ideal_j,max_j);
    create_ideal(generators_j,count_j,semigroup,g_s,ideal_j);
    g_j = find_frobenius(ideal_j, max_j);
    n_j = count_elements(ideal_j, g_j);
    system ("cls");
    cout << "\n\nJ = ";
    print_array(ideal_j, g_j);
```

```
    print_other(g_j, n_j);
    system ("Pause");
    ideal_choice = toupper(ideal_choice_menu());
    continue;
   }  //End of Ideal Choice B

//*******************              Compute I+J          *******************

   if (ideal_choice == 'C')
   {
    if (flags[2] != 1 || flags[3] != 1)  // Check if I & J are Computed
    {
     ideal_choice = toupper(ideal_error());
     continue;
    }
    flags[4]= 1;
    count_ij = count_i * count_j;   //Number of generators for I+J
    initialize_array(generators_ij, count_ij);
    add_ideals(generators_ij, generators_i, count_i, generators_j, count_j);
    max_ij = g_s + generators_ij[0]+1;
    initialize_array(ideal_ij,max_ij);
    create_ideal(generators_ij,count_ij,semigroup,g_s,ideal_ij);
    g_ij = find_frobenius(ideal_ij, max_ij);
    n_ij = count_elements(ideal_ij, g_ij);
    system ("cls");
    cout << "\n\nI+J = ";
    print_array(ideal_ij, g_ij);
    print_other(g_ij, n_ij);
    system ("Pause");
    ideal_choice = toupper(ideal_choice_menu());
    continue;
   }  //End of Ideal Choice C

//*******************              Compute I-J          *******************

   if (ideal_choice == 'D')
   {
    if (flags[2] != 1 || flags[3] != 1)  //Check if I & J are Computed
    {
     ideal_choice = toupper(ideal_error());
     continue;
    }
    flags[5]= 1;
    flags[11]=1;
    min_dij = generators_i[0] - generators_j[0];
    max_dij = g_i - generators_j[0] + generators_j[count_j-1];
```

```
    if (min_dij < 0)
    {
     cout << "The dual contains negative numbers.\n";
     cout << "Please adjust I and/or J.\n";
    }
    else
    {
     create_dual(ideal_i, g_i, generators_j, count_j, dual_ij, min_dij, max_dij);
     system ("cls");
     cout << "\n\nI-J = ";
     print_array(dual_ij, max_dij);
    }
    system ("Pause");
    ideal_choice = toupper(ideal_choice_menu());
    continue;
    flags[11]=0;
   }  // End of ideal choice D

//*******************            Compute J-I          *******************

   if (ideal_choice == 'E')
   {
    if (flags[2] != 1 || flags[3] != 1)  //Check if I & J are Computed
    {
     ideal_choice = toupper(ideal_error());
     continue;
    }
    flags[6]= 1;
    flags[11]=1;
    min_dji = generators_j[0] - generators_i[0];
    max_dji = g_j - generators_i[0] + generators_i[count_i-1];

    if (min_dji < 0)
    {
     cout << "The dual contains negative numbers.\n";
     cout << "Please adjust I and/or J.\n";
    }
    else
    {
     create_dual(ideal_j, g_j, generators_i, count_i, dual_ji, min_dji,
                max_dji);
     system ("cls");
     cout << "\n\nJ-I = ";
     print_array(dual_ji, max_dji);
    }
```

```
    system ("Pause");
    ideal_choice = toupper(ideal_choice_menu());
    continue;
    flags[11]=0;
   }  // End of ideal choice E

//*******************            Compute S-I          *******************

   if (ideal_choice == 'F')
   {
    if (flags[0] != 1 || flags[2] != 1)  //Check if S & I are Computed
    {
     ideal_choice = toupper(ideal_error());
     continue;
    }
    flags[7]= 1;
    flags[11]=1;
    min_dsi = generators_s[0] - generators_i[0];
    max_dsi = g_s - generators_i[0] + generators_i[count_i-1];

    if (min_dsi < 0)
    {
     cout << "The dual contains negative numbers.\n";
     cout << "Please adjust I and/or S.\n";
    }
    else
    {
     create_dual(semigroup, g_s, generators_i, count_i, dual_si, min_dsi,
                 max_dsi);
     system ("cls");
     cout << "\n\nS-I = ";
     print_array(dual_si, max_dsi);
    }
    system ("Pause");
    ideal_choice = toupper(ideal_choice_menu());
    continue;
    flags[11]=0;
   }  // End of ideal choice F

//*******************            Compute S-J          *******************

   if (ideal_choice == 'G')
   {
    if (flags[0] != 1 || flags[3] != 1)  //Check if S & J are Computed
    {
     ideal_choice = toupper(ideal_error());
```

```
     continue;
    }
    flags[8]= 1;
    flags[11]=1;
    min_dsj = generators_s[0] - generators_j[0];
    max_dsj = g_s - generators_j[0] + generators_j[count_j-1];

    if (min_dsj < 0)
    {
     cout << "The dual contains negative numbers.\n";
     cout << "Please adjust I and/or J.\n";
    }
    else
    {
     create_dual(semigroup, g_s, generators_j, count_j, dual_sj, min_dsj,
                 max_dsj);
     system ("cls");
     cout << "\n\nS-J = ";
     print_array(dual_sj, max_dsj);
    }
    system ("Pause");
    ideal_choice = toupper(ideal_choice_menu());
    continue;
    flags[11]=0;
   }  // End of ideal choice G

//*******************          Compute (I+J)-I        *******************

   if (ideal_choice == 'H')
   {
    if (flags[2] != 1 || flags[3] != 1)  //Check if I & J are Computed
    {
     ideal_choice = toupper(ideal_error());
     continue;
    }
    if (flags[4] ==0)
    {
     cout << "You must compute I+J first.\n";
     ideal_choice = toupper(ideal_error());
     continue;
    }
    min_dij_i = generators_ij[0] - generators_i[0];
    max_dij_i = g_ij - generators_i[0] + generators_i[count_i-1];

    if (min_dij_i < 0)
    {
```

```
   cout << "The dual contains negative numbers.\n";
   cout << "Please adjust I and/or J.\n";
  }
  else
  {
   flags[11]=1;
   create_dual(ideal_ij, g_ij, generators_i, count_i, dual_ij_i,
               min_dij_i, max_dij_i);
   system ("cls");
   cout << "\n\n(I+J)-I = ";
   print_array(dual_ij_i, max_dij_i);
  }
  system ("Pause");
  ideal_choice = toupper(ideal_choice_menu());
  continue;
  flags[11]=0;
 }  // End of ideal choice H

//*******************          Compute (I+J)-J        *******************

 if (ideal_choice == 'I')
 {
  if (flags[2] != 1 || flags[3] != 1)  //Check if I & J are Computed
  {
   ideal_choice = toupper(ideal_error());
   continue;
  }
  if (flags[4] ==0)
  {
   cout << "You must compute I+J first.\n";
   ideal_choice = toupper(ideal_error());
   continue;
  }
  min_dij_j = generators_ij[0] - generators_j[0];
  max_dij_j = g_ij - generators_j[0] + generators_j[count_j-1];

  if (min_dij_j < 0)
  {
   cout << "The dual contains negative numbers.\n";
   cout << "Please adjust I and/or J.\n";
  }
  else
  {
   flags[11]=1;
   create_dual(ideal_ij, g_ij, generators_j, count_j, dual_ij_j,
               min_dij_j, max_dij_j);
```

```
     system ("cls");
     cout << "\n\n(I+J)-J = ";
     print_array(dual_ij_j, max_dij_j);
    }
    system ("Pause");
    ideal_choice = toupper(ideal_choice_menu());
    continue;
    flags[11]=0;
   }  //End of Ideal Choice I

//*******************       Print the Elements of S        *******************

  if (ideal_choice == 'J')
  {
   if (flags[0] != 1)
   {
    cout << "\n\n";
    ideal_choice = toupper(enter_s_error());
    continue;
   }
   system ("cls");
   cout << "\n\nS = ";
   print_array(semigroup, g_s);
   print_other(g_s, n_s);
   system ("Pause");
   system ("cls");
   ideal_choice = toupper(ideal_choice_menu());
   continue;
  }   //End of Ideal Choice J

//*******************       Print the Elements of I        *******************

  if (ideal_choice == 'K')
  {
   if (flags[2] != 1)  //Check if I are Computed
     {
     ideal_choice = toupper(ideal_error());
     continue;
     }
   system ("cls");
   cout << "\n\nI = ";
   print_array(ideal_i, g_i);
   print_other(g_i, n_i);
   system ("Pause");
   system ("cls");
   ideal_choice = toupper(ideal_choice_menu());
```

```
   continue;
  }   //End of Ideal Choice K

//*******************     Print the Elements of J     *******************

  if (ideal_choice == 'L')
  {
   if (flags[3] != 1)  //Check if J are Computed
     {
      ideal_choice = toupper(ideal_error());
      continue;
     }
   system ("cls");
   cout << "\n\nJ = ";
   print_array(ideal_j, g_j);
   print_other(g_j, n_j);
   system ("Pause");
   system ("cls");
   ideal_choice = toupper(ideal_choice_menu());
   continue;
  }   //End of Ideal Choice L

    cout << "\nYou entered an invalid letter. Try again:\n\n";
    system ("Pause");
    ideal_choice = toupper(ideal_choice_menu());   //display menu
  }  //End of Ideal Choice Menu

   cout << "\n";
   system ("Pause");
   system ("cls");
   main_choice = toupper(main_choice_menu());  //return to main menu
   continue;
 }  //End of Main Choice C

//*******************       Go to Apery Menu      *******************

  if (main_choice == 'D')
  {
   if (flags[0] != 1)       //verify S is alreay computed
   {
    main_choice = toupper(enter_s_error());  //error and get new choice
    continue;
   } //end error

   apery_choice = toupper(apery_choice_menu());   //display menu
   while (apery_choice != 'X')       //while != return to main
```

```
  {
   flags[10] = 1;
//*******************          Compute Apery (Ap)      *******************

   if (apery_choice == 'A')
   {
    flags[9] = 1;
    max_ap = g_s + generators_s[0]  + 1;
    initialize_array(apery, max_ap);
    create_apery(generators_s, semigroup, g_s, apery, max_ap);
    system ("cls");
    cout << "\n\nAp(S)=";
    print_array(apery, max_ap);
    system ("Pause");
    apery_choice = toupper(apery_choice_menu());
    continue;
   }  //End Apery Choice A

//*******************          Compute Ap'          *******************

   if (apery_choice == 'B')
   {
    if (flags[9] != 1)  //Check if Apery Set is Computed
    {
     apery_choice = toupper(apery_error());
     continue;
    }  //end error
    initialize_array(apery_prime, max_ap);
    create_prime(semigroup, apery, max_ap, apery_prime);
    system ("cls");
    cout << "\n\nAp'(S)=";
    print_array(apery_prime, max_ap);
    system ("Pause");
    apery_choice = toupper(apery_choice_menu());
    continue;
   }   //End of Apery Choice B

//*******************          Compute Ap*          *******************

   if (apery_choice == 'C')
   {
    if (flags[9] != 1)  //Check if Apery Set is Computed
     {
      apery_choice = toupper(apery_error());
      continue;
     }  //end error
```

```
    initialize_array(apery_star, max_ap);
    create_star(semigroup, apery, max_ap-1, apery_star);
   system ("cls");
   cout << "\n\nAp*(S)=";
    print_array(apery_star, max_ap);
    system ("Pause");
    apery_choice = toupper(apery_choice_menu());
    continue;
  }   //End of Apery Choice C

  cout << "\nYou entered an invalid letter. Try again:\n\n";
  system ("Pause");
  apery_choice = toupper(apery_choice_menu());   //display menu
 }// End Apery while loop
 flags[10]= 0;
 system ("Pause");
 system ("cls");
 main_choice = toupper(main_choice_menu());  //return to main menu
 continue;
}    //End of Main Choice D


//*******************      Print the Elements of S       *******************

if (main_choice == 'E')
{
 if (flags[0] != 1)
 {
  cout << "\n\n";
  main_choice = toupper(enter_s_error());
  continue;
 }
 system ("cls");
 cout << "\n\nS = ";
 print_array(semigroup, g_s);
 print_other(g_s, n_s);
 system ("Pause");
 system ("cls");
 main_choice = toupper(main_choice_menu());
 continue;
}   //End of Main Choice E


cout << "\nYou entered an invalid letter. Try again:\n\n";
system ("Pause");
system ("cls");
```

```
  main_choice = toupper(main_choice_menu());

 }  //End of Main Choice Menu

cout << "\n";
system("PAUSE");
return 0;
}
//   End of Main Program, begin Functions


///////////////////////////           Main Choice Menu          //////////////////////////////////

char main_choice_menu(void)         //Main choice menu function
{

  char choice;     //letter selected from welcome menu

  cout << "Main Menu:  Please select what you would like to do.\n\n"
      << "A - Enter generators and compute the numerical semigroup, S.\n\n"
      << "B - Compute H(S), the holes of S.\n\n"
      << "C - Ideals...\n\n"
      << "D - Apery...\n\n"
      << "E - Display the elements of S.\n\n"
      << "X - Leave this world.\n\n";
  cin >> choice;
  return choice;
}

////////////////////////          Choice Menu for Ideals          ///////////////////////////////
char ideal_choice_menu(void)
{
  char choice;

  system ("cls");

  cout << "\nIdeal Menu: Please select what you would like to do.\n\n"
      << "A - Enter generators and compute the ideal 'I'.\n"
      << "B - Enter Generators adn compute the ideal J.\n"
      << "C - Compute 'I+J'.\n"
      << "D - Compute the dual 'I-J'.\n"
      << "E - Compute the dual 'J-I'.\n"
      << "F - Compute the dual 'S-I'.\n"
      << "G - Compute the dual 'S-J'.\n"
      << "H - Compute (I+J)-I.\n"
      << "I - Compute (I+J)-J.\n"
```

```
     << "J - Display the elements of S.\n"
     << "K - Display the elements of I.\n"
     << "L - Display the elements of J.\n"
     << "X - Return to Main Menu.\n\n";

  cin >> choice;
  return choice;
}


////////////////////////          Choice Menu for Apery          ///////////////////////////////
char apery_choice_menu(void)
{
  char choice;

  system ("cls");

  cout << "Apery Menu:  Please select what you would like to do. \n\n"
     << "A - Compute Ap\n\n"
     << "B - Compure Ap'\n\n"
     << "C - Compute Ap*\n\n"
     << "X - Return to Main Menu.\n\n";

  cin >> choice;
  return choice;
}


//////////////////////////          Initial Int Arrays          ////////////////////////////////
void initialize_array(int array[], int count)
{
  int i;      //index for loop
  for (i=0; i<count; i++)
    array[i]= 0;
}


//////////////////////          Enter S First Error Message     ///////////////////////////
char enter_s_error()
{
  char choice;

  cout << "You need to compute S first.\n\n";
  system ("Pause");
  system ("cls");
```

```
  choice = main_choice_menu();
  return choice;
}


//////////////////////      Create the Holes of S       ///////////////////////
void create_holes(int h[], int s[], int gs)
{
  int i,n;

  for (i=0; i<gs; i++)
  {
    if (s[i]==1) continue;
    n = gs-i;
    if (s[n]==1) continue;
    h[i]=1;
    h[n]=1;
  }
}


//////////////////////    Missing Ideal Data Error Message   ///////////////////////
char ideal_error()
{
  char choice;

  cout << "Insufficient data to perform this calculation.\n\n";
  system ("Pause");
  system ("cls");

  choice = ideal_choice_menu();
  return choice;
}


//////////////////////Missing Apery Data Error Message//////////////////////
char apery_error()
{
  char choice;

  cout << "Insufficient data to perform this calculation.\n\n";
  system ("Pause");
  system ("cls");

  choice = apery_choice_menu();
  return choice;
```

```
}


//////////////////////////        Number of Generators        ////////////////////////////////
int many_generators()
{
 int count;
 printf("\nHow many generators do you want to enter? ");
 scanf("%i", &count);
 return count;
}


///////////////////////////          Get the Generators        //////////////////////////////////
void get_generators( int gen[], int count)
{
 for (i=0; i<count; i++)
 {
  printf("Enter Generator # %i:  ", i+1);
  scanf("%i", &gen[i]);
 }
}


///////////////////////          Include the Generators       //////////////////////////////////
void include_gen(int gen[], int count, int array[])
{
 int i,n;

 for (i=0; i<count; i++)
 {
  n = gen[i];
  array[n] = 1;
 }
}


///////////////////////////          Create the Semigroup    ///////////////////////////////
void create_s(int gen[], int count, int group[], int maximum)
{
 int i,j,k;

 group[0] = 1;  //include 0 in the semigroup
 //loop through the semigroup array starting at m(S)+1
 for (i=gen[0]+1; i<maximum; i++)
 {
```

```
    if (group[i] ==1)  continue;   //determine if already in the group
    //Loop through generators to see if (element - generator) in group
    for (j=0; j<count; j++)
    {
     k = i - gen[j];
     if (group[k]==1)     //if element - generate is in group
     {
      group [i] = 1;     //element is in group
      break;
     }
    }
  }
}


///////////////////////////        Create an Ideal        ////////////////////////////////////
void create_ideal(int gen[], int count, int group[], int g_s, int ideal[])
{
  int i,j,k;

  for (i=0; i<count; i++)
  {
    for (j=0; j<g_s; j++)
    {
     if (group[j] == 0) continue;
     k = j + gen[i];
     ideal[k] = 1;
    }
  }
}


///////////////////////////       Add Two Ideals          ////////////////////////////////
void add_ideals(int sum_ideal[], int gen_1[], int count_1, int gen_2[],
              int count_2)
//determine the generators of the sum of two ideals

{
  int i,j,k,n;

  n=0;
  for (i=0; i<count_1; i++)
  {
    for (j=0; j<count_2; j++)
    {
     k = gen_1[i]+gen_2[j];
```

```
    sum_ideal[n] = k;
    n++;
   }
  }
 cout << "\n";
}


///////////////////////////        Create Dual of Two Arrays    ////////////////////////
void create_dual(int array_1[], int g_1, int gen_2[], int count_2, int dual[],
                int minimum, int maximum)
{
 int i,j,n;

 initialize_array(dual, maximum);
 for (i=minimum; i<=maximum; i++)
 {
  for (j=0; j<count_2; j++)
  {
   n = gen_2[j] + i;
   if (n > g_1)
   {
    dual[i] = 1;
    j = count_2;
    continue;
   }
   if (array_1[n] == 0)
   {
    dual[i] = 0;
    j = count_2;
   }
   else
    dual[i] = 1;
  }
 }
}


///////////////////////////     Find Frobenius Number      /////////////////////////////
int find_frobenius(int group[], int g)
{
 g--;
 while (group[g] != 0) g--;
 return g;
}
```

```
//////////////////////////      Count Number of Elements    ////////////////////////////
int count_elements(int array[], int count)
{
  int i;   //index for loop.
  int n=0;   // count number of elements in the array
  for (i=0; i<count; i++)
   if (array[i] == 1)
    n++;
  return n;
}


///////////////////////         Print the Array           //////////////////////////////////
void print_array(int array[], int count)
{
  cout << "{";
  for (i=0; i<count; i++)
   if (array[i] == 1)
     cout << i << " ";
     if (flags[1]!=1 && flags[10]!=1 && flags[11]!=1)
     cout << count+1 << "...";
     if (flags[11]==1) cout << count << "...";

     cout << "}\n\n";
}


//////////////////////    Print Frobenius and Number of Elements     /////////////////
void print_other(int g, int n)
{
  cout << "The Frobenius Number is " << g;
  cout << "\n\nThe number of elements is " << n << "\n\n";
}


////////////////////////////     Create the Apery Set      ////////////////////////////
void create_apery(int gen[], int group[], int g_s, int apery[], int count)
{

  int i,n;

  apery[0] = 1;
  for (i=0; i<=count; i++)
   {
   n = i - gen[0];
```

```
   if (n<0) continue;
   if (group[i] == 1 && group[n] == 0)
     apery[i] = 1;
   if (i>g_s && group[n] == 0)
     apery[i] = 1;
  }
}


//////////////////////////////     Create AP'     //////////////////////////////////////
void create_prime(int group[], int apery[], int max_ap, int prime[])
{
  int i,j,n;

  prime[max_ap-1] = 1;

  for (i=max_ap; i>0; i--)
  {
    if (apery[i] == 0) continue;   //do not check if not in Ap
    for (j=max_ap; j>i; j--)
    {
      if (apery[j] == 0) continue;
      n=j-i;
      if (group[n] == 0)
      {
        prime[i] = 1;
        j=i;
      }
    }
  }
}


////////////////////////////     Creat Ap*     ////////////////////////////////////////
void create_star(int group[], int apery[], int maximum, int star[])
{
  int i, n;

  for (i=0; i<maximum; i++)
  {
    if (apery[i] == 0)  continue;
    n = maximum - i;
    if (group[n] == 0) star[i] = 1;
  }
}
```

```
////////////////////////////  Flags  ////////////////////////////////////////
/* flags[0] = generators of S
   flags[1] = H(S)
   flags[2] = I
   flags[3] = J
   flags[4] = I+J
   flags[5] = I-J
   flags[6] = J-I
   flags[7] = S-I
   flags[8] = S-J
   flags[9] = Ap
   flags[10] = Ap print
   flags[11] = dual print


*/
```

REFERENCES

[1] R. Apery, *Sur les branches superlineaires des courbes algebriques*, C.R. Acad. Sci. Paris **222** (1946)

[2] V. Barucci, D. Dobbs and M. Fontana, *Maximality properties in numerical semigroups and applications to one-dimensional analytically irreducible local domains*, Memoirs of the American Mathematical Society **125** (1997)

[3] V. Barucci and R. Froberg, *One-dimensional almost Gorenstein rings*, J. Algebra **188** (1997), 418-442

[4] P. Constapel, *The product of an integrally closed ideal with its inverse*, Comm. Algebra **27** (1999), 3777 - 3779

[5] R. Froberg, C. Gottlieb and R. Haggkvist, *On numerical semigroups*, Semigroup Forum **35** (1987), 63 - 83

[6] K. Herzinger, *Torsion in the Tensor Product of an ideal with its inverse*, Comm. Algebra **24** (1996), 3065-3083

[7] E. Kunz, *The value-semigroup of a one-dimensional Gorenstein ring*, Proc. Amer. Math. Soc. **25** (1970), 748-751

[8] J.C. Rosales, P.A. Garcia-Sanchez, J.I. Garcia-Garcia and M.B. Branco, *Systems of inequalities and numerical semigroups*, J. London Math. Soc. **65** (2002) 611-623

[9] K. Rosen, *Elementary number theory and its applications*, 4^{th} edition, Addison Wesley Longman, Reading, Massachusetts, 2000