**Project Title:**

* **“CANDY COMPANY SCHEME MANAGEMENT SYSTEM”**

**Project Description:**

**Overview:**

* The Candy Company Scheme Management System is a tool for managing employees that mimics the structure of a multi-level marketing company using binary trees. In this system, each employee can recruit up to two subordinates (one on the left and one on the right). It also keeps track of their candy collections and gold earnings.

**Objective:**

* To create an efficient system that:
  1. Manages a hierarchical employee structure
  2. Tracks reward distribution (candy and gold)
  3. Automates the calculation of benefits based on recruitment
  4. Maintains data persistence across sessions
  5. Provides an easy-to-use interface for employee management

**Scope:**

* The project focuses on managing a simple company hierarchy with these specific boundaries:  
  1. A binary tree structure where each employee can have a maximum of two direct subordinates.
  2. Basic employee data storage (name, gold amount, candy count)
  3. Simple reward system where upper management benefits from lower-level recruitment
  4. Core operations: hire, remove, search, and display employees
  5. File-based data storage for maintaining records between sessions
  6. A console-based interface with menu-driven options for easy navigation.

**Features:**

A.) **Employee Management System**

* Enables creation and management of hierarchical employee structures, including hiring new employees and assigning them to specific positions under existing employees.

B.) **Automatic Reward Distribution**

* Implements an upward-flowing reward system where recruiting new employees triggers candy distribution to all superiors in the chain, with each candy automatically converting to 20 gold.

C.) **Data Persistence**

* Maintains employee records, hierarchy relationships, and reward statistics in a text file database, ensuring data survives between program sessions.

D.) **Search Functionality**

* Provides capability to search for specific employees and view their current status, including gold and candy counts.

E.) **Employee Removal System**

* Allows removal of employees with built-in safeguards to prevent orphaned subordinates and maintain CEO position integrity.

**Technologies Used:**

* **Programming Language:** C programming Language
* **Frameworks/Libraries:** <stdio.h>, <stdlib.h>, <string.h>, <unistd.h>
* **Tools**: VS Code, AI, Git
* **Database:** Notepad

**Project Structure:**

CC214L/

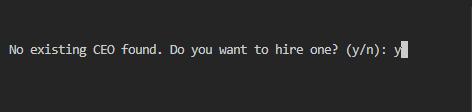
├── Main.c # Main Program

├── Main.exe # Executable file of main

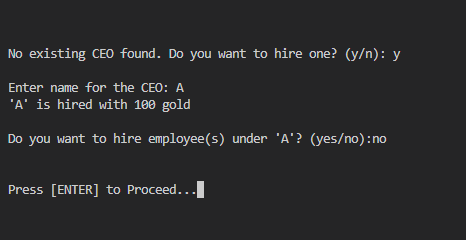
├── README.md # Project description

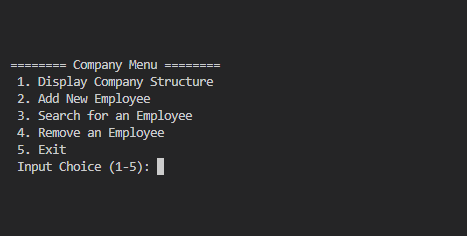
└── candyDatabase.txt # Data storage file for the company structure

**Usage:**

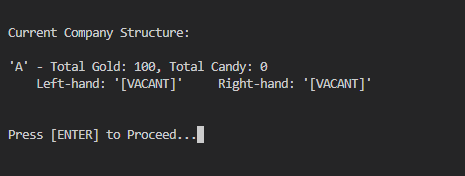
If the database is empty, the user will hire a CEO

After hiring a CEO, it will ask if the user wants to hire an employee under the CEO.   
This can go on until the user is satisfied.

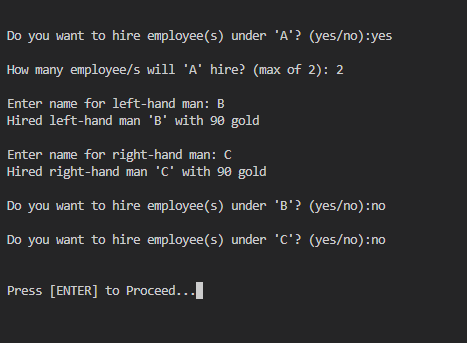


If the database is not empty, it will automatically display the menu.

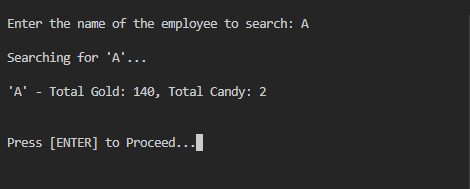
If option 1 is selected, it will display the data inside the database. If an employee or CEO does not   
have any employee working under them, it will be labeled as VACANT.



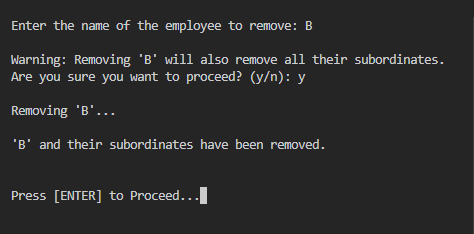
If option 2 is selected, the program asks the user about hiring employees under them. Of course, the user can say no if satisfied.

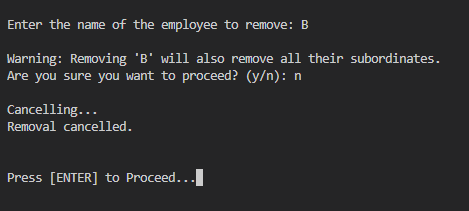


If option 3 is selected, the program will ask the user a name to search and the program then navigates the name and displays it along with its total gold and candies.

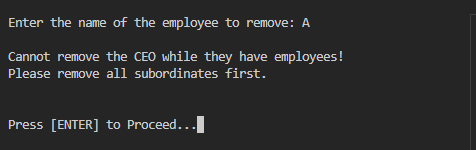


If option 4 is selected, the program will ask the user a name to remove, the program then navigates the name and remove the node along with its subordinates or employees working under them.

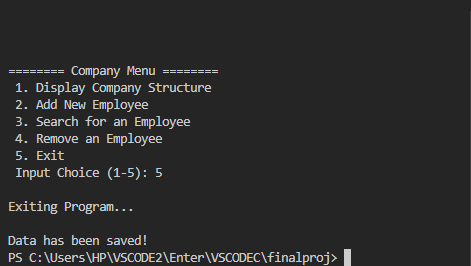


Of course, the user can say no.

However, if the CEO is about to be removed and still have employees, it will be not removed.   
The subordinates must be removed first.



If option 5 is selected, the program just simply terminates.



**Testing:**

**Setup Testing**

1. Launch the program
2. Verify loading screen appears
3. Check if database file exists or new CEO creation is prompted

**Functionality Tests**

**Employee Management**

1. Add new CEO
   * Enter CEO name
   * Verify 100 gold is assigned
   * Check if saved to database
2. Add Subordinates
   * Add left-hand employee
   * Add right-hand employee
   * Verify gold decreases by 10 for each level
   * Confirm candy distribution upward

**Search Function**

1. Enter existing employee name
   * Verify correct gold and candy amounts shown
2. Enter non-existent name
   * Verify appropriate message shown

**Removal Testing**

1. Remove regular employee
   * Check if subordinates are removed
   * Verify database update
2. Test CEO removal restrictions
   * Attempt CEO removal with subordinates
   * Verify protection message

**Data Persistence**

1. Add employees
2. Close program
3. Relaunch
4. Verify structure remains intact

**Acknowledgements:**

**Project Team**

* Marco Montellano
* Fanny Rose Caballes
* Benjie Sumile
* Angel Mae Santosidad
* Hazzel Canama
* Vincent Bernabe Romeo
* Justin Dumon

**Academic Support**

* BSIT 2A CC214L Data Structures and Algorithms Course

**Technical Resources**

* C Standard Library
  + stdio.h – printf(), scanf(), getchar()
  + stdlib.h – malloc(), free()
  + string.h – strcasecmp(), strcpy()
  + unistd.h – usleep(), sleep()

**Resources:**

* Educational online materials on binary tree structures.
  + GeeksforGeeks
  + Programiz
  + W3Schools
  + Javatpoint
  + GitHub

**Special Thanks**Special appreciation to our professor and fellow leaders of the group who guided us throughout the development of this Data Structures and Algorithms final project.

**Contact Information:**

* **Author:** Marco Montellano
* **E-mail:** [crmchs.montellano.marco@gmail.com](mailto:crmchs.montellano.marco@gmail.com)
* **GitHub:** https://github.com/macoy147