UNIVERSITY OF TECHNOLOGY, JAMAICA

Faculty Of Engineering and Computing

School Of Computing & Information Technology

CMP2006 - DATA STRUCTURES

Programming Group Project

2023-24, Summer Semester

Tutor: Alesia Bowen-Mighty

Group:

Stephen Burnett, Breanna Smith, Geovana Lewis, Iyana Taylor

AMCS 2

Due: July 29, 2024

Documentation 1, Point 1: Details of each member's contribution

Group Member	Contribution
Stephen Burnett	Player class, Card class, Worst case asymptotic analysis
Breanna Smith	Puzzle class, Worst case asymptotic analysis
Geovana Lewis	Node class, Wheel class, Worst case asymptotic analysis, Discussion
Iyana Taylor	Queue class, Game class, Worst case asymptotic analysis, Discussion, User Manual, File writing

Documentation 1, Point 2: Discuss which data structure(s) were used for relevant components and why

1. **String vector in the Puzzle class:**
    - Used for storing possible solutions and display states of the puzzle efficiently, allowing easy access and modification of elements.
2. **Character vector in the Wheel class:**

- Utilized to represent the sections of the wheel, facilitating random selection and easy updates when needed.

3. **Player vector in the Game class:**
   - Employed to maintain a list of players, enabling easy iteration and access to player data during the game.
4. Circular Linked List in the Wheel Class:
   - Used to mimic the behaviour of the physical wheel spun in the game.
5. Queue:
   - Used to track the guessed letters and bought vowels in each round to ensure players do not try to guess or buy a consonant or vowel repeatedly.
6. Node structure in the Queue class:
   - Used to declare queue attributes and create the node constructor without creating another node class.

The vector data structures were used for their dynamic sizing and efficient access and modification properties.

Documentation 1, Point 3: Declaration of Authorship for each member

UNIVERSITY OF TECHNOLOGY, JAMAICA
Declaration of Authorship

FACULTY:______Engineering and Computing_________________________

School/Department: ____School of Computing and Information Technology_______

Course Code & Title: ________________ CMP2006 Data Structures ____________

Submitted To:________Mrs. Alesia Bowen-Mighty____________________
(Lecturer/Supervisor)

Submitted By: _________Breanna Smith__________________________
Student's name

____________2207630____
ID Number

___________________Green Island District, Greeen Island P.O, Hanover.___________________________
Address

_________________(876) 480-9163____
Contact telephone numbers (home, work, cell)

Date of Submission: _____________July 9, 2024_________________________

Title of Assignment: ___________Data Structures Group Project______________

____

****

Declaration: I certify that I am the author of this paper and that any assistance I received in its preparation is fully acknowledged and disclosed in the paper. I have also cited all sources from which I used visuals, data, ideas or words, either quoted directly or paraphrased. I also certify that this paper was prepared by me specifically for this course. I also understand that a grade will not be assigned without the submission of this agreement.

Student's Signature:_____________B.Smith_____________________________________

****

Lecturer's/Supervisor's Grade for Assignment: ___________________________

Lecturer's/Supervisor's Comments: ____

____

Note: For group assignments each student is required to complete a separate Declaration of Authorship.

Ref: Regulation 5*: Conditions and Procedures*

Governing Student Academic Misconduct

Division of Student Services & Registry

Ac. Brd. Sept 23, 2004

UNIVERSITY OF TECHNOLOGY, JAMAICA
Declaration of Authorship

FACULTY: Faculty of Engineering and Computing

School/Department: School of Computing and Information Technology

Course Code & Title: CMP2006-Data Structures

Submitted To: Mrs. Alesia Bowen-Mighty
(Lecturer/Supervisor)

Submitted By: Stephen Burnett
Student's name

2210182
ID Number

Mount Salem, Montego Bay, St.James
Address

876-229-2189
Contact telephone numbers (home, work, cell)

Date of Submission: July 29,2024

Title of Assignment: Data Structures Group Project

****

Declaration: I certify that I am the author of this paper and that any assistance I received in its preparation is fully acknowledged and disclosed in the paper. I have also cited all sources from which I used visuals, data, ideas or words, either quoted directly or paraphrased. I also certify that this paper was prepared by me specifically for this course. I also understand that a grade will not be assigned without the submission of this agreement.

Student's Signature: S.Burnett

****

Lecturer's/Supervisor's Grade for Assignment: ____________

Lecturer's/Supervisor's Comments: ____________

Note: For group assignments each student is required to complete a separate Declaration of Authorship.

***Ref: Regulation 5**: Conditions and Procedures Governing Student Academic Misconduct*

Division of Student Services & Registry
Ac. Brd. Sept 23, 2004

UNIVERSITY OF TECHNOLOGY, JAMAICA
Declaration of Authorship

FACULTY: Faculty of Engineering and Computing

School/Department: School of Computing and Information Technology

Course Code & Title: CMP2006 - Data Structures

Submitted To: Mrs. Alesia Bowen-Mighty
(Lecturer/Supervisor)

Submitted By: Geovana Lewis
Student's name

2210184
ID Number

Cousins Cove, Lucea P.O, Hanover
Address

876-278-2937
Contact telephone numbers (home, work, cell)

Date of Submission: July 29,2024

Title of Assignment: Data Structures Group Project

____

****

Declaration: I certify that I am the author of this paper and that any assistance I received in its preparation is fully acknowledged and disclosed in the paper. I have also cited all sources from which I used visuals, data, ideas or words, either quoted directly or paraphrased. I also certify that this paper was prepared by me specifically for this course. I also understand that a grade will not be assigned without the submission of this agreement.

Student's Signature: [illegible]

****

Lecturer's/Supervisor's Grade for Assignment: ________________________

Lecturer's/Supervisor's Comments: ________________________________

____

Note: For group assignments each student is required to complete a separate Declaration of Authorship.

***Ref: Regulation 5**: Conditions and Procedures Governing Student Academic Misconduct*

UNIVERSITY OF TECHNOLOGY, JAMAICA
Declaration of Authorship

FACULTY: Faculty of Engineering and Computing

School/Department: School of Computing and Information Technology

Course Code & Title: CMP2006 - Data Sructures

Submitted To: Alesia Bowen-Mighty
(Lecturer/Supervisor)

Submitted By: Iyana Taylor
Student's name

2209566
ID Number

Burnt Ground, Ramble P.O., Hanover
Address

876-213-8437
Contact telephone numbers (home, work, cell)

Date of Submission: July 29, 2024

Title of Assignment: Data Structures Programming Group Project 2023-2024

****

Declaration: I certify that I am the author of this paper and that any assistance I received in its preparation is fully acknowledged and disclosed in the paper. I have also cited all sources from which I used visuals, data, ideas or words, either quoted directly or paraphrased. I also certify that this paper was prepared by me specifically for this course. I also understand that a grade will not be assigned without the submission of this agreement.

Student's Signature: IR. Taylor

****

Lecturer's/Supervisor's Grade for Assignment: ____________________

Lecturer's/Supervisor's Comments: ____________________

Note: For group assignments each student is required to complete a separate Declaration of Authorship.

***Ref: Regulation 5**: Conditions and Procedures Governing Student Academic Misconduct*

Documentation 1, Point 4: Worst Case Asymptotic Analysis of the key features/methods used in the system

Player Class

1. Player() : playernumber(0), name(""), grandtotal(0), roundtotal(0) {}
    - playernumber = 0 → O(1)
    - name = "" → O(1)
    - grandtotal = 0 → O(1)
    - roundtotal = 0 → O(1)
2. Player(int number, const string& playername) : playernumber(number), name(playername), grandtotal(0), roundtotal(0) {}
    - playernumber = number → O(1)
    - name = playername → O(1)
    - grandtotal = 0 → O(1)
    - roundtotal = 0 → O(1)
3. Player(const Player& other) : playernumber(other.playernumber), name(other.name), grandtotal(other.grandtotal), roundtotal(other.roundtotal) {}
    - playernumber = other.playernumber → O(1)
    - name = other.name → O(1)
    - grandtotal = other.grandtotal → O(1)

- roundtotal = other.roundtotal → O(1)

4. int getplayernumber() const { return playernumber; }
   - return playernumber → O(1)
5. string getname() const { return name; }
   - return name → O(1)
6. int getgrandtotal() const { return grandtotal; }
   - return grandtotal → O(1)

7. int getroundtotal() const { return roundtotal; }
   - return roundtotal → O(1)
8. void setplayernumber(int number) { playernumber = number; }
   - playernumber = number → O(1)
9. void setname(const string& playername) { name = playername; }
   - name = playername → O(1)
10. void addroundtotal(int amount) { roundtotal += amount; }
    - roundtotal += amount → O(1)
11. Add to grand total (version 1) void addtograndtotal() { grandtotal += roundtotal; }
    - grandtotal += roundtotal → O(1)

12. Add to grand total (version 2) void addtograndtotal(int amount) { grandtotal += amount; }
    - grandtotal += amount → O(1)
13. Reset round total void resetroundtotal() { roundtotal = 0; }
    - roundtotal = 0 → O(1)
14. Reset grand total void resetgrandtotal() { grandtotal = 0; }
    - grandtotal = 0 → O(1)

Category for Player Class

For Player class, each method and constructor call has a time complexity of O(1). Therefore, the overall worst-case time complexity for operations in these classes is:

O(Player)= O(1)

Card Class

1. Default constructor Card() : type(MONEY), value(0) {}
    - type = MONEY → O(1)
    - value = 0 → O(1)
2. Card(CardType cardType, int cardValue) : type(cardType), value(cardValue) {}
    - type = cardType → O(1)
    - value = cardValue → O(1)

3. Copy constructor Card(const Card& other) : type(other.type), value(other.value) {}
    - type = other.type → O(1)
    - value = other.value → O(1)
4. CardType getType() const { return type; }
    - return type → O(1)
5. int getValue() const { return value; }
    - return value → O(1)
6. void setType(CardType cardType) { type = cardType; }
    - type = cardType → O(1)
7. void setValue(int cardValue) { value = cardValue; }
    - value = cardValue → O(1)

Category for Card Class

Since all operations in the Card class are constant time, the overall time complexity is:

O(Card)=O(1)

Puzzle Class

1. loadPuzzle ()
    - O (n) - Linear
    - The time required to complete the function grows linearly with the size of the input string

2. guessLetter ()
    - O (n) – Linear
    - Method to check if the guessed letter entered matches the puzzle grows linearly
3. revealPuzzle ()
    - O(n^2) – Quadratic
    - Method has an overall time complexity because the loop iterates over each character of the string puzzle, so it runs (n) times and grows linear
4. isSolved ()
    - O(n^k) - Polynomial
    - Method is a combination two linear time
5. getCategory()
    - 0(1) – Constant
    - Returns the value of a member variable to ‘category’

Wheel Class

1. initializeWheel()
        - O(n^2) – Quadratic
        - multiple calls to addCard, each of which has a linear time complexity
2. addCard()

  - O(n) – Linear
  - Needs to traverse the entire list to find the last node before adding the new node.

3. spinWheel()
    - O(1) – Constant
    - performs a fixed number of iterations, bounded by a small, constant range, 50 to 100   (in other words the loop is bounded by a constant maximum value of 100)
4. getCurrentCard()
    - O(1) – Constant
    - Method simply returns a value

Queue Class

1. **clear()**
    - O(n)
    - Requires dequeuing all elements in the queue.
2. **enqueue(char letter)**
    - O(1)
    - Always adds a new node at the end of the queue.
3. **dequeue()**
    - O(1)
    - Always removes the front node from the queue.

4. **isEmpty() const**
    - O(1)
    - Simply checks if the front pointer is NULL.
5. **peek() const**
    - O(1)
    - Returns the data from the front node without removing it.
6. **display() const**
    - O(n)
    - Iterates through all elements to display them.
7. **contains(char c) const**
    - O(n)
    - Searches through the entire queue to find the character.

Game Class

Though ‘n’ is commonly used to represent input size in Big O analysis, to distinguish all the different parameters and input sizes different letters will be used since multiple factors affect complexity (number of players, length of puzzle, number of actions per turn, size of guessed letters in the queue, number of rounds). Though the worst-case asymptotic analysis will be given using ‘n’ as well.

Let:

n: Represent the number of lines (puzzles) in the file.

m: Represent the length of the puzzle.

g: Represent the size of the guessed letters queue.

p: Represent the number of players.

a: Represent the number of actions within a turn.

r: Represent the number of rounds.

1. **loadPuzzlesFromFile(const string& filename)**: O(n)
   - Reading n lines from a file.
   - O(n)
2. **countOccurrences(const string& str, char letter)**: O(m)
   - Counting occurrences of a character in a string (puzzle) of length m.
   - O(n)
3. **advanceToNextPuzzle()**: O(1)
   - Advancing to the next puzzle and resetting the guessed letters queue.
   - O(1)
4. **guessLetter(Player& player, int cardValue)**: O(m + g)
   - Counting occurrences in the puzzle (O(m)) and queue operations (O(g)), where m is the length of the puzzle and g is the size of the guessed letters queue.
   - O(2n) = O(n)
5. **buyVowel(Player& player)**: O(m + g)

    - Similar to guessLetter with the addition of vowel purchase logic.
    - O(2n) = O(n)

6. **solvePuzzle(Player& player, int cardValue)**: O(m)
    - Comparing the solved attempt with the puzzle.
    - O(n)
7. **checkWinner()**: O(p)
    - Checking the grand total for each player, where p is the number of players.
    - O(n)
8. **handlePlayerTurn(Player& player)**: O(a * (m + g))
    - Handling player actions, where a is the number of actions within a turn.
    - O(n * (2n)) = $O(2n^2) = O(n^2)$
9. **startGame()**: O(n)
    - Loading puzzles and initializing the first round.
    - O(n)
10. **playRounds()**: O(r * (p * a * (m + g)))
    - Playing r rounds, each involving p players, each taking a turn with a actions.
    - O(n * (n * n * (n + n))) = O(n * (n * n * (2n))) = $O(n * (n^2 * 2n)) = O(n * 2n^3)$

$= O(2n^4) = O(n^4)$

11. **playRound()**: O(p * a * (m + g))
    - Similar to handlePlayerTurn but iterates through all players in a round.
    - $O(n * n * (n + n)) = O(n * n * (2n)) = O(n^2 * (2n)) = O(2n^3) = O(n^3)$
12. **endGame()**: O(p)
    - Resetting data and checking if the game should end or proceed.
    - O(n)

Write_Puzzles

- O(1).

Developer Notes, Improvements and Suggestions

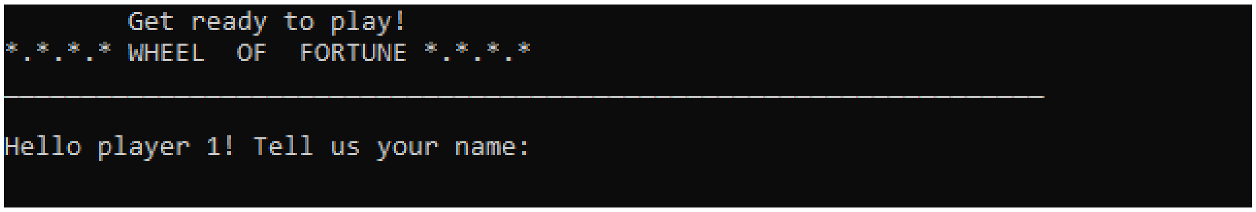
1. **Consistent Error Handling**: Employ more exception handling techniques throughout all class methods, for consistency.
2. **Input Validation**: Validate user inputs in all places where necessary.
3. **Separation of Concerns**: Break down some larger methods into smaller, more manageable helper functions to improve readability and maintainability.
4. **Documentation**: Adding comments and documentation for each method to help in understanding the code better at first glance and maintaining it in the future.

5. **Puzzle Cycling**: Adjust the game code to cycle through all puzzles in the file, and to restart at the top when all puzzles have been used. The game currently restarts at the very first puzzle in the file for every new game (set of 3 rounds). So, only the first three puzzles are being used. A circular linked list may be used.

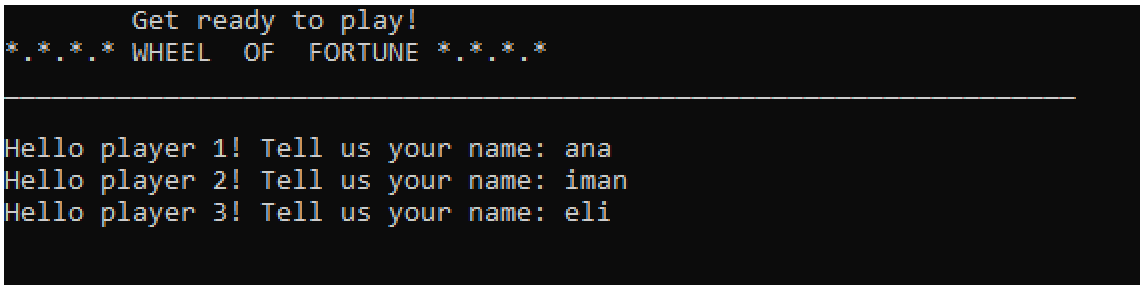
6. **Bonus Round**: Wheel of Fortune usually has a bonus round where the winner of the preceding 3 rounds can try at solving an extra puzzle to gain additional money or win a prize. That bonus round could be implemented in the game during future development as an extension.

Documentation 2: Application User Manual

1. Upon starting the application, there will be a prompt to enter the names of the 3 players:



```
        Get ready to play!
*.*.*.* WHEEL  OF  FORTUNE *.*.*.*
___________________________________________________________________

Hello player 1! Tell us your name:
```



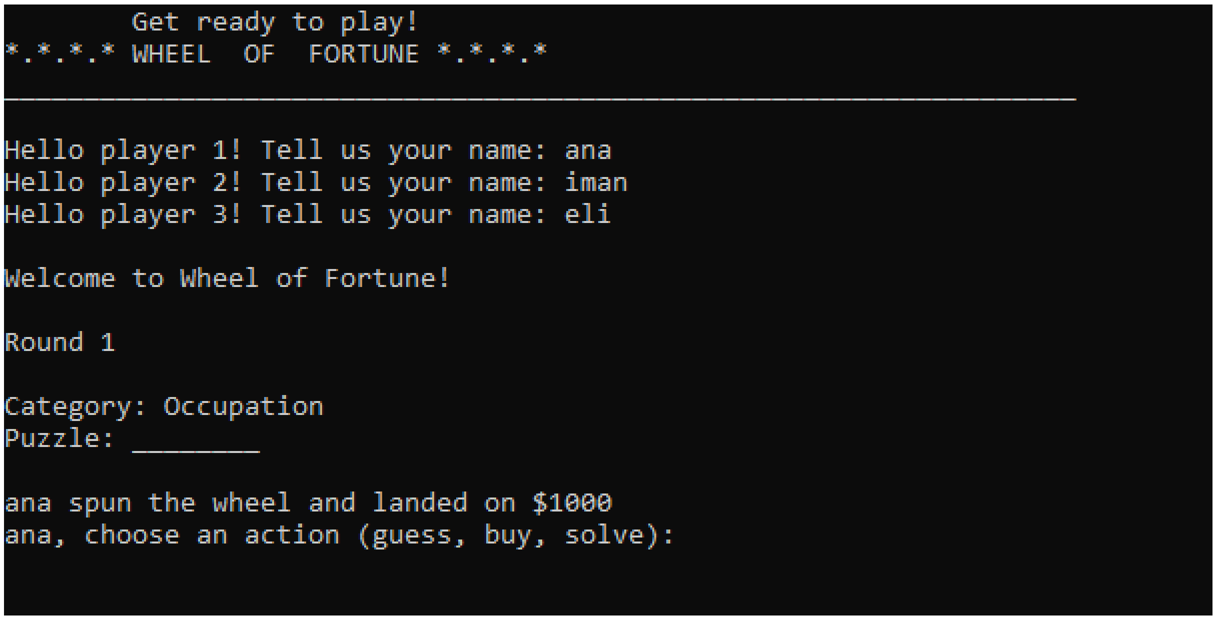
```
        Get ready to play!
*.*.*.* WHEEL  OF  FORTUNE *.*.*.*
___________________________________________________________________

Hello player 1! Tell us your name: ana
Hello player 2! Tell us your name: iman
Hello player 3! Tell us your name: eli
```

2. After which a greeting will come up, the category and initial puzzle state displayed, and player 1 will spin the wheel (spinning is handled by the application) and will be prompted to choose an action by typing “guess”, “buy”, or “solve”.

   To note:

   - Only consonants can be guessed, this action is free and is a way to earn money in each round.
   - “Buy” means to purchase vowels, a vowel costs $50 (in the Wheel of Fortune gameshow vowels cost $250 but $50 seems more accessible to novice players).
   - If a player feels they know the puzzle answer they may solve freely by typing the entire answer (with punctuations and spaces). Solving does not cost money. Players can also earn in rounds by solving the puzzles.

- Spinning on “BANKRUPT” means the player loses all their money and their turn. Spinning on “LOSE TURN” means the player simply loses their turn.



```
        Get ready to play!
*.*.*.* WHEEL  OF  FORTUNE *.*.*.*
___________________________________________________________________

Hello player 1! Tell us your name: ana
Hello player 2! Tell us your name: iman
Hello player 3! Tell us your name: eli

Welcome to Wheel of Fortune!

Round 1

Category: Occupation
Puzzle: ________

ana spun the wheel and landed on $1000
ana, choose an action (guess, buy, solve):
```

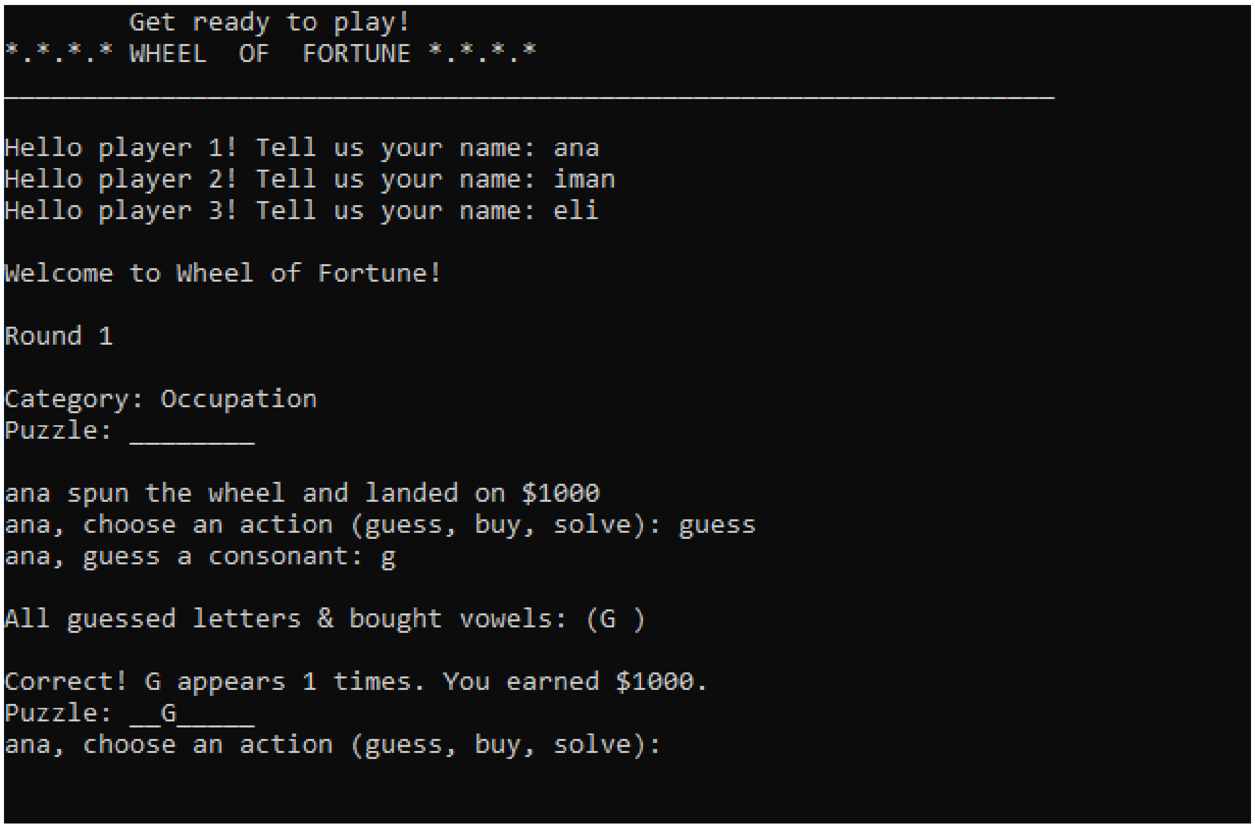
3. After choosing an action, the player will be prompted to enter their choice of letter to guess, vowel to buy or their solve attempt. Guessed letters and bought vowels (if the purchase can be made) will be displayed in a queue so all players are aware and will not repeat those letters.

   To note:

   - If the player enters a non-alphabet guess, a message will be displayed saying it is invalid. They will be asked to guess again.
   - A message will be displayed for incorrect letter guesses.
   - A message will be displayed if players do not have enough money to buy vowels.
   - A message will be displayed if the solve attempt is incorrect.

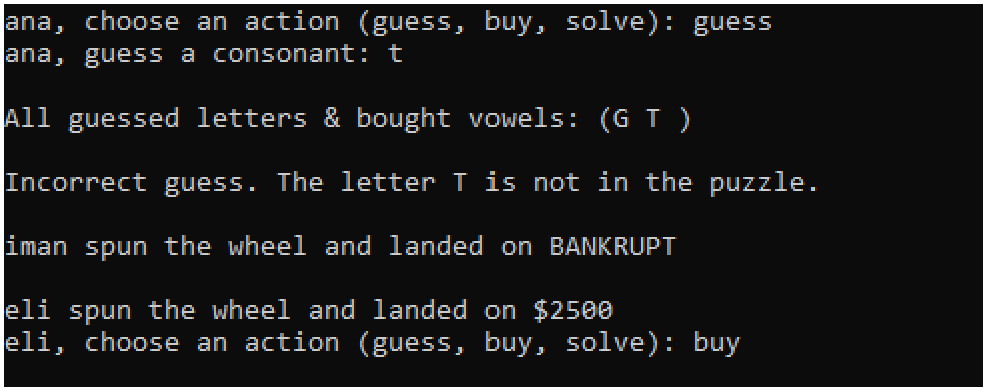
- If the guessed letter or bought vowel is correct, the player will be told so. The number of times the letter appears in the puzzle will be multiplied by the value of the card spun on and the result added to the player's round total.

- If a guess is correct, that player may continue to guess (until they make an incorrect one).

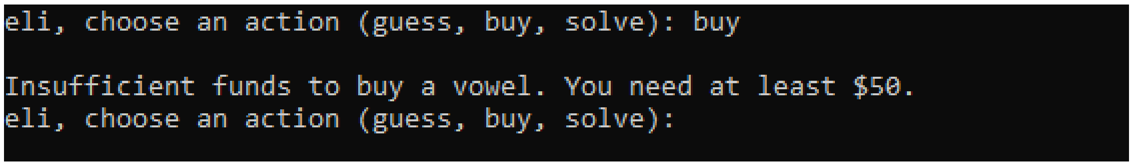


```
        Get ready to play!
*.*.*.* WHEEL  OF  FORTUNE *.*.*.*
___________________________________________________________________

Hello player 1! Tell us your name: ana
Hello player 2! Tell us your name: iman
Hello player 3! Tell us your name: eli

Welcome to Wheel of Fortune!

Round 1

Category: Occupation
Puzzle: ________

ana spun the wheel and landed on $1000
ana, choose an action (guess, buy, solve): guess
ana, guess a consonant: g

All guessed letters & bought vowels: (G )

Correct! G appears 1 times. You earned $1000.
Puzzle: __G_____
ana, choose an action (guess, buy, solve):
```

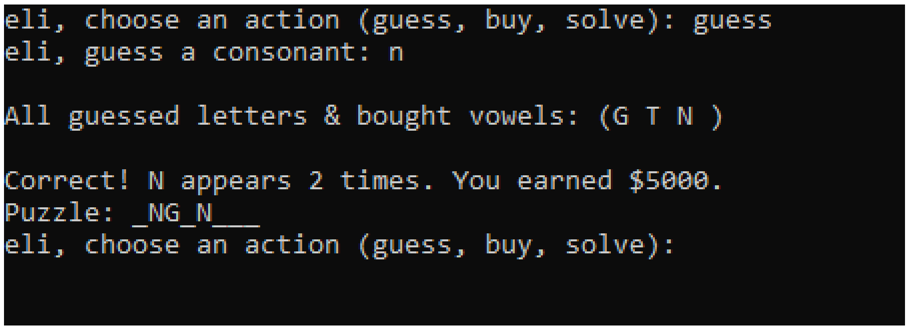
4. See in the image below, a player guesses incorrectly, the turn moves to the next player. Unfortunately, player 2 spins on "BNAKRUPT" losing their money and turn. The turn moves to player 3.


```
ana, choose an action (guess, buy, solve): guess
ana, guess a consonant: t

All guessed letters & bought vowels: (G T )

Incorrect guess. The letter T is not in the puzzle.

iman spun the wheel and landed on BANKRUPT

eli spun the wheel and landed on $2500
eli, choose an action (guess, buy, solve): buy
```

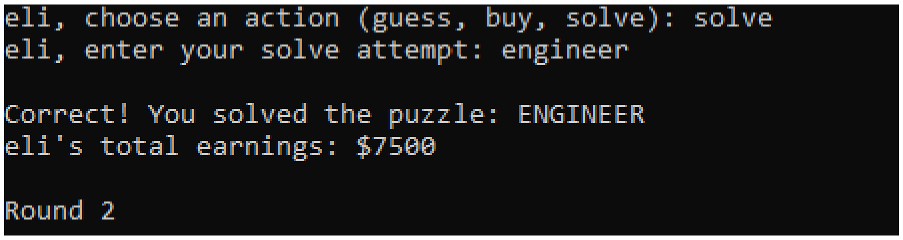
5. Player 3 chooses to buy a vowel, but they don't have enough to make a purchase. The application lets them know of such and prompts them to choose an action again.


```
eli, choose an action (guess, buy, solve): buy

Insufficient funds to buy a vowel. You need at least $50.
eli, choose an action (guess, buy, solve):
```

6. They choose to guess and is prompted for the guess.


```
eli, choose an action (guess, buy, solve): guess
eli, guess a consonant: n

All guessed letters & bought vowels: (G T N )

Correct! N appears 2 times. You earned $5000.
Puzzle: _NG_N___
eli, choose an action (guess, buy, solve):
```

7. Player 3 continues after a correct guess and chooses to solve, the solve attempt is correct and the money they've earned for the round displayed. After the puzzles is solved (either through guessing, buying or solving) the game advances to the next round. The same procedures continue for the next 3 rounds and at the end of round 3, the winner is declared based on the player with the highest total. The application will ask if players would like to play again.



```
eli, choose an action (guess, buy, solve): solve
eli, enter your solve attempt: engineer

Correct! You solved the puzzle: ENGINEER
eli's total earnings: $7500

Round 2
```