Monopoly

Major Project

This document is the entire written work of both team members, contains all necessary documentation including: IPO Charts, Flowcharts, Detailed Algorithms for each game module, as well as detailed program instructions. It also includes the meetings the group has had, Gantt charts, task lists and schedules.

2015 12SDD

Wood6, Peter

Year 12 Software Design and Development

2015 12SDD

Contents

[Instructions 4](#_Toc427864938)

[Rules 4](#_Toc427864939)

[Object 5](#_Toc427864940)

[Preparation 5](#_Toc427864941)

[Banker 5](#_Toc427864942)

[The Bank 5](#_Toc427864943)

[The Play 5](#_Toc427864944)

[Game Modules 8](#_Toc427864945)

[Main Menu 8](#_Toc427864946)

[IPO 8](#_Toc427864947)

[Flowchart 8](#_Toc427864948)

[Algorithm 8](#_Toc427864949)

[Start Turn 9](#_Toc427864950)

[IPO 9](#_Toc427864951)

[Flowchart 9](#_Toc427864952)

[Algorithm 9](#_Toc427864953)

[End Turn 10](#_Toc427864954)

[IPO 10](#_Toc427864955)

[Flowchart 10](#_Toc427864956)

[Algorithm 10](#_Toc427864957)

[Auction 11](#_Toc427864958)

[IPO 11](#_Toc427864959)

[Flowchart 11](#_Toc427864960)

[Algorithm 11](#_Toc427864961)

[Bid 12](#_Toc427864962)

[Pass 12](#_Toc427864963)

[Forfeit 12](#_Toc427864964)

[Building Management 13](#_Toc427864965)

[IPO 13](#_Toc427864966)

[Flowchart 14](#_Toc427864967)

[Algorithm 15](#_Toc427864968)

[Add House 15](#_Toc427864969)

[Remove House 15](#_Toc427864970)

[Add Hotel 15](#_Toc427864971)

[Remove Hotel 15](#_Toc427864972)

[Mortgage 16](#_Toc427864973)

[Un-mortgage 16](#_Toc427864974)

[Roll Dice 17](#_Toc427864975)

[IPO 17](#_Toc427864976)

[Flowchart 17](#_Toc427864977)

[Algorithm 17](#_Toc427864978)

[Trade 18](#_Toc427864979)

[IPO 18](#_Toc427864980)

[Flowchart 18](#_Toc427864981)

[Algorithm 18](#_Toc427864982)

[Jail 19](#_Toc427864983)

[IPO 19](#_Toc427864984)

[Flowchart 19](#_Toc427864985)

[Flowchart 20](#_Toc427864986)

[Algorithm 21](#_Toc427864987)

[Purchase Property 22](#_Toc427864988)

[IPO 22](#_Toc427864989)

[Flowchart 22](#_Toc427864990)

[Algorithm 22](#_Toc427864991)

[Task List 23](#_Toc427864992)

[Sam 23](#_Toc427864993)

[Peter 23](#_Toc427864994)

[Collaboration 23](#_Toc427864995)

[Group Meetings 24](#_Toc427864996)

[24/06/2015 24](#_Toc427864997)

[Assessment 24](#_Toc427864998)

[Date issued 24](#_Toc427864999)

[Due date 24](#_Toc427865000)

[Work Conducted 24](#_Toc427865001)

[Encountered Problems 24](#_Toc427865002)

[26/06/2015 25](#_Toc427865003)

[Assessment 25](#_Toc427865004)

[Date issued 25](#_Toc427865005)

[Due date 25](#_Toc427865006)

[Work Conducted 25](#_Toc427865007)

[Encountered Problems 25](#_Toc427865008)

[04/07/2015 26](#_Toc427865009)

[Assessment 26](#_Toc427865010)

[Date issued 26](#_Toc427865011)

[Due date 26](#_Toc427865012)

[Work Conducted 26](#_Toc427865013)

[Encountered Problems 26](#_Toc427865014)

[15/07/2015 27](#_Toc427865015)

[Assessment 27](#_Toc427865016)

[Date issued 27](#_Toc427865017)

[Due date 27](#_Toc427865018)

[Work Conducted 27](#_Toc427865019)

[Encountered Problems 27](#_Toc427865020)

[17/07/2015 28](#_Toc427865021)

[Assessment 28](#_Toc427865022)

[Date issued 28](#_Toc427865023)

[Due date 28](#_Toc427865024)

[Work Conducted 28](#_Toc427865025)

[Encountered Problems 28](#_Toc427865026)

[24/07/2015 29](#_Toc427865027)

[Assessment 29](#_Toc427865028)

[Date issued 29](#_Toc427865029)

[Due date 29](#_Toc427865030)

[Work Conducted 29](#_Toc427865031)

[Encountered Problems 29](#_Toc427865032)

[06/08/2015 30](#_Toc427865033)

[Assessment 30](#_Toc427865034)

[Date issued 30](#_Toc427865035)

[Due date 30](#_Toc427865036)

[Work Conducted 30](#_Toc427865037)

[Encountered Problems 30](#_Toc427865038)

[18/08/2015 31](#_Toc427865039)

[Assessment 31](#_Toc427865040)

[Date issued 31](#_Toc427865041)

[Due date 31](#_Toc427865042)

[Work Conducted 31](#_Toc427865043)

[Encountered Problems 31](#_Toc427865044)

[20/08/2015 32](#_Toc427865045)

[Assessment 32](#_Toc427865046)

[Date issued 32](#_Toc427865047)

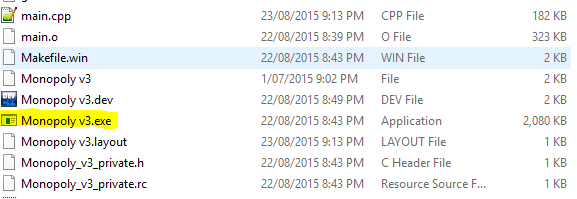
[Due date 32](#_Toc427865048)

[Work Conducted 32](#_Toc427865049)

[Encountered Problems 32](#_Toc427865050)

[Schedule 33](#_Toc427865051)

# Instructions

Welcome to Monopoly, as created by Sam and Peter. To run the game you simply open the ‘Monopoly v3.exe’ file. As shown below:

## Rules

The game uses standard ‘Classic’ Monopoly rules.

### Object

The object of the game is to become the wealthiest player through buying, renting and selling property.

### Preparation

As this is a digital game, no preparation is needed. Each player starts with $1500 in their wallet and no properties. The Chance and Community Chest cards are shuffled.

### Banker

No banker is needed as the computer does all required banking.

### The Bank

There is no defined ‘Bank’, instead the computer handles all required banking, and it also holds the title deeds for properties, the houses and hotels. The Bank can never go broke.

### The Play

The player’s each roll the dice in turn, and the player with the highest total starts the play. All player’s start on ‘GO’. The player who rolled the highest chooses how they would like to start the turn. They may choose to go bankrupt, which returns all of their property to the bank and skips them in succeeding turns. They may choose to trade with another player, or they may manage property. When the player decides to roll the dice their token is moved the required amount of spaces.

According to the space that the token lands on, the player may have to pay money to the bank or another player, draw a card from either Community Chest or Chance, go to jail, or they may be able to buy property.

If the player rolls two of the same number, referred to as doubles, they get to roll again for a second turn. This can be repeated up to three times in a row, however if three doubles are thrown in a row by the same player, they are sent to jail.

#### GO

When a player lands on or passes the ‘GO’ tile, they are paid $200 by the Bank. A player may only be awarded this salary once each time they go around the board, however if they receive the Community Chest card or Chance card that says “Advance to GO”, they may collect their salary again.

#### Buying Property

Whenever the player lands on an un-owned property they may buy that property from the Bank at its printed price. They receive the Title Deed card showing ownership. If they do not wish to buy the property, it is auctioned to the players and given to the highest bidder. The buyer pays the Bank the amount of the bid and receives the Title Deed card for that property. Any player, including the one who declined the option to buy it at the printed price, may bid. Bidding may start at any price.

#### Paying Rent

When the player lands on property owned by another player, the owner collects rent from them in accordance with the list printed on its Title Deed card. If the property is mortgaged, no rent can be collected. It is an advantage to hold all the Title Deed cards in a colour group because the owner may then charge double rent for unimproved properties in that colour-group. It is even more advantageous to have houses or hotels on properties because rents are much higher than for unimproved properties.

#### Chance and Community Chest

When a player lands on either of these spaces, a card is drawn from the corresponding array of cards (deck), the instructions on the card are followed and the card is then placed at the bottom of the ‘deck’. If ‘Get out Of Jail Free’ is the card drawn, it is held by the player until they use it, and it is then returned to the deck.

#### Income Tax

If the player lands on this space, they have two options. They may pay an estimated amount of $200, or they may pay 10% of their total worth. The total worth is your wallet, combined with the value of properties, houses and hotels.

#### Jail

There are three conditions for landing in jail:

1. Landing on ‘Go To Jail’.
2. Throwing three doubles in a row.
3. Drawing a card that says ‘Go to Jail’.

When a player is sent to Jail, they cannot collect salary for passing ‘GO’, however they do still collect rent. If the player lands on the ‘Jail’ tile, then they are ‘Just Visiting’, and no penalty is incurred. To get out of jail the player must do one of the following:

1. Throw doubles while in jail.
2. Use a ‘Get Out Of Jail Free’ Card.
3. Pay for another players ‘Get out Jail Free’ Card.
4. Pay the $50 fine.

#### Free Parking

If any player lands on ‘Free Parking’, nothing happens. This tile is merely a free space on the board.

#### Houses

When all properties in a colour group are owned, houses may be built on them to improve their rent. Houses must be bought from the Bank, and only a limited number of houses are available (32).

When building houses or hotels, they must be built on properties evenly. I.e. The player cannot build a second house on a property until each property in the colour group has one house on it. When removing houses and hotels, they must also be broken down evenly.

#### Hotels

When a player has four houses built on each property in the colour group, they may build a hotel. There are twelve hotels available in the game.

#### Building Shortages

When the Bank has no houses or hotels left, the player wishing to build must wait for another player to return their buildings to the bank.

#### Selling Property

Unimproved properties, railroads and utilities may be sold to any player in a trade. To trade properties, no buildings can be on any other property in that colour group.

#### Mortgages

Unimproved properties can be mortgaged at the start or end of any turn. Any buildings on the property must be removed before mortgaging.

No rent is collected on mortgaged properties, but rent is collected on other un-mortgaged properties in the colour group.

To lift the mortgage, the owner must pay the printed price on the title deed.

#### Bankruptcy

You are declared bankrupt if you owe more than you can pay to the Bank or another player. All of your properties and houses are returned to the bank.

#### To Win

In order to win, you must be the only player left in the game. All other players must be bankrupt.

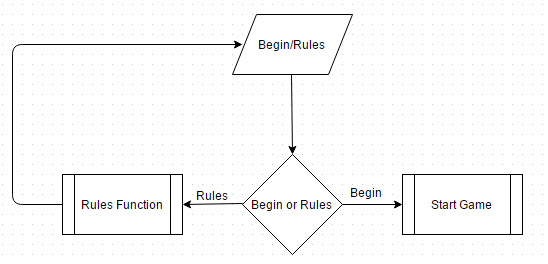
# Game Modules

## Main Menu

### IPO

|  |  |  |
| --- | --- | --- |
| ***Input*** | ***Process*** | ***Output*** |
| Begin | Start game |  |
| Rules | Call rules function |  |

### Flowchart



### Algorithm

Ask for input

If they entered R for rules

Call Rules function

If they entered B to begin the game

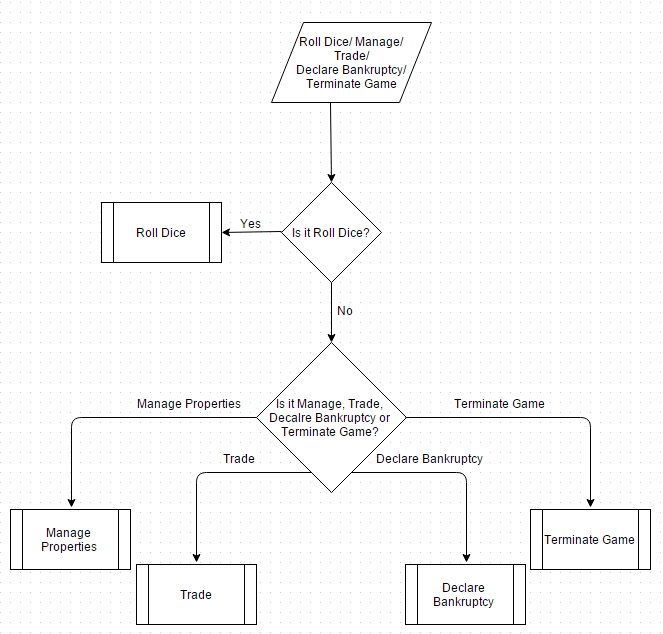
End function and return to main function to start

## Start Turn

### IPO

|  |  |  |
| --- | --- | --- |
| ***Input*** | ***Process*** | ***Output*** |
| Roll dice | Call roll dice function | Two random numbers between 1 and 6 as ASCII dice |
| Manage | Call manage function | Properties owned |
| Trade | Call trade function | List of players that can trade |
| Declare bankruptcy | Call declare bankruptcy function | Player is now bankrupt |
| Terminate game | Call terminate game function | Game ending |

### Flowchart



### Algorithm

Output user options

Ask for input

Determine if input was valid, if it is valid continue, otherwise repeat again

Determine action to be taken depending on input

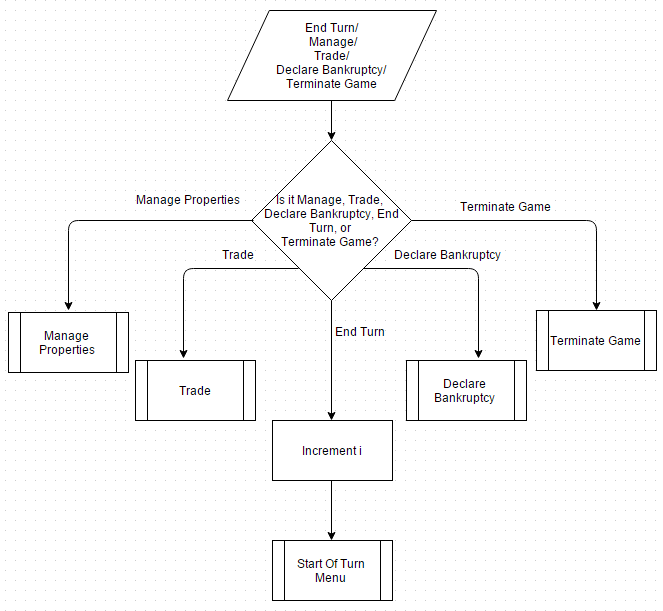
Call appropriate function (trade, manage, declare bankruptcy, terminate game, roll dice)

## End Turn

### IPO

|  |  |  |
| --- | --- | --- |
| ***Input*** | ***Process*** | ***Output*** |
| End turn | Call stat of turn function for next i | Next players turn |
| Manage | Call manage function | Properties owned |
| Trade | Call trade function | List of players that can trade |
| Declare bankruptcy | Call declare bankruptcy function | Player is now bankrupt |
| Terminate game | Call terminate game function | Game ending |

### Flowchart



### Algorithm

Output user options

Ask for input

Determine if input was valid, if it is valid continue, otherwise repeat again

Determine action to be taken depending on input

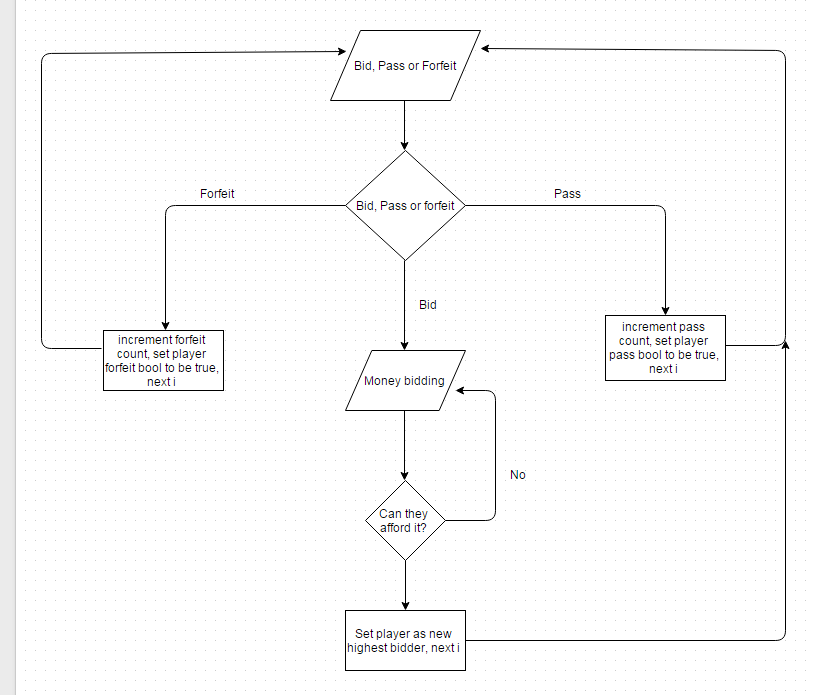
Call appropriate function (trade, manage, declare bankruptcy, terminate game, end turn)

## Auction

### IPO

|  |  |  |
| --- | --- | --- |
| ***Input*** | ***Process*** | ***Output*** |
| Bid   * Money Bidding | Determine if they can afford their bid | New highest bid |
| Pass | Increment Pass count for the turn | Player passed |
| Forfeit | Increment the forfeit count, set player forfeit boll to be true | Player forfeited |

### Flowchart



### Algorithm

Check if the current player has forfeited, if not then continue, if they have then go to the next iteration

Output the property being auctioned, the current players wallet and the highest bid

Determine if player’s wallet is less than the current bid

If it isn’t, call forfeit function and skip to end of function

Ask for player input determining what action they would like to take

Call appropriate function below

Iterate through players UNTIL all but one forfeit, OR all but one pass

The player with the highest bid is given the property

Subtract the bid from their wallet

### Bid

#### Algorithm

Ask for input determining the player’s bid

Determine if input is higher than the current bid, and less than their wallet value

If it is less than highest bid, output that the bid must be higher than the current bid

If their bid exceeds wallet value, output that the bid must be lower

Set highest bid to be the player’s bid

### Pass

#### Algorithm

Set the player passed bool to be true

Increment pass count for the turn

### Forfeit

#### Algorithm

Set the player forfeit bool to be true

Increment forfeit count

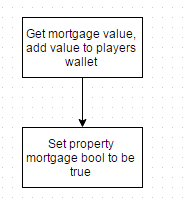
## Building Management

### IPO

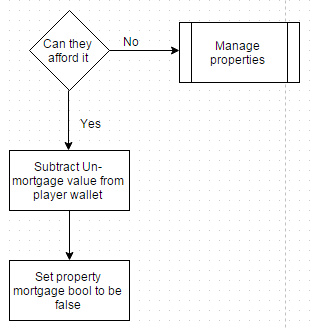
|  |  |  |
| --- | --- | --- |
| ***Input*** | ***Process*** | ***Output*** |
| Property being managed | call appropriate function |  |
| * mortgage | add mortgage value to the player wallet, set property mortgage bool to be true | Remaining amount in wallet. |
| * Un-mortgage | Subtract un-mortgage value from the player wallet, set property mortgage bool to be false | Remaining amount in wallet. |
| * Add house | get value of house, can player afford house, is player building evenly, increment house count, decrement houses left, subtract cost from players wallet | Remaining amount in wallet, amount of houses left. |
| * Add hotel | get value of hotel, can player afford hotel, is player building evenly, increment house count, decrement hotels left, subtract cost from wallet | Remaining amount in wallet, amount of hotels and houses left. |
| * Remove house | get value of house, divide value by 2, determine if player is removing evenly, decrement house count, increment houses left, add value to players wallet | Remaining amount in wallet, amount of houses left. |
| * Remove hotel | get value of hotel, divide by 2, is player removing evenly, decrement house count, increment hotels left, add value to wallet | Remaining amount in wallet, amount of hotels and houses left. |

### Flowchart

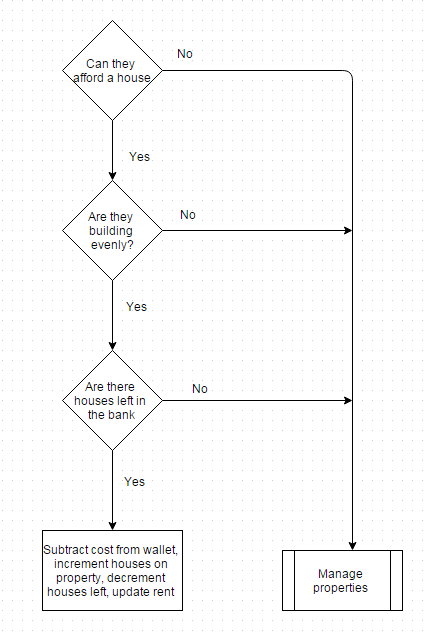
#### Mortgage



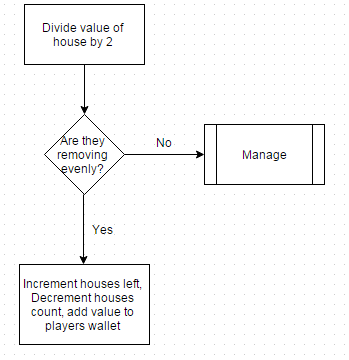
#### Un-mortgage



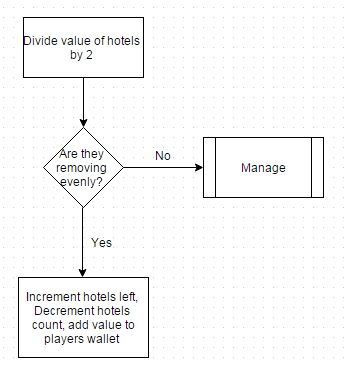
#### Add House



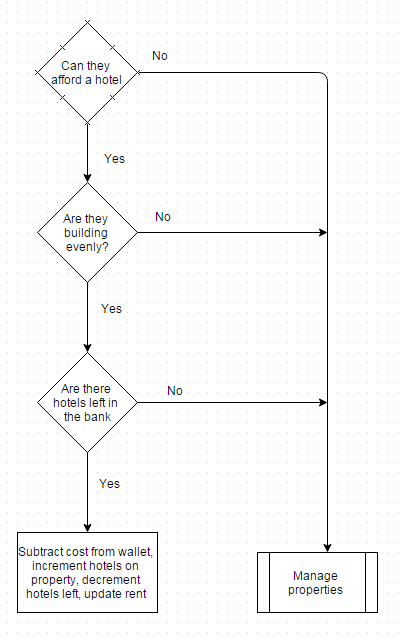
#### Remove House



#### Remove Hotel



#### Add Hotel



### Algorithm

Output the properties owned by the player

Ask for input that determines what property is going to be managed

Display options for that property

Ask for input determining what action is to be taken

Call appropriate function below

### Add House

#### Algorithm

Get the value of a house for the selected property

Determine if the player can afford a house, if not then terminate function

Determine if the player is building evenly, if not then terminate function

Increment the house count on that property

Decrement the houses left

Subtract cost from player wallet

Return to previous screen

### Remove House

#### Algorithm

Get the value of a house for the selected property

Divide the value by 2

Determine if the player is removing evenly, if not then terminate function

Decrement the house count on that property

Increment the houses left

Add value to player wallet

Return to previous screen

### Add Hotel

#### Algorithm

Get the value of a hotel for the selected property

Determine if the player can afford a hotel, if not then terminate function

Determine if the player is building evenly, if not then terminate function

Increment the house count on that property

Decrement the hotels left

Subtract cost from player wallet

Return to previous screen

### Remove Hotel

#### Algorithm

Get the value of a hotel for the selected property

Divide the value by 2

Determine if the player is removing evenly, if not then terminate function

Decrement the house count on that property

Increment the hotels left

Add value to player wallet

Return to previous screen

### Mortgage

#### Algorithm

Get the mortgage value of the property

Set the property mortgaged bool to be true

Add value to player wallet

Return to previous screen

### Un-mortgage

#### Algorithm

Get the un-mortgage cost of the property

Set the property mortgaged bool to be false

Subtract cost from player wallet

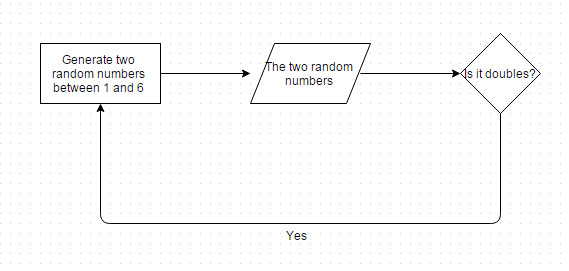
Return to previous screen

## Roll Dice

### IPO

|  |  |  |
| --- | --- | --- |
| ***Input*** | ***Process*** | ***Output*** |
| N/A | Generate two numbers between 1 and 6, store numbers | Output numbers |

### Flowchart



### Algorithm

Generate Random Number between 1 and 6

Store that number

Generate another Random Number between 1 and 6

Store that number

Output the appropriate combination of dice to the screen.

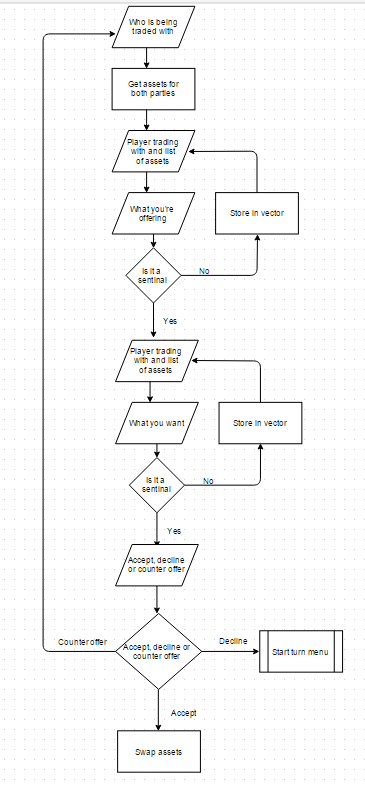
Return the Sum of the two dice to the main function.

## Trade

### IPO

|  |  |  |
| --- | --- | --- |
| Input | Process | Output |
| Who is to be traded with | Get assets for both parties | List of assets |
| What you’re offering | Store in vector | Assets being traded and offered |
| What you want | Store in different vector | Assets being traded and offered |
| Accept | Swap assets | N/A |
| Decline | terminate | N/A |
| Counter | Start function again | Assets for both parties |

### Flowchart



### Algorithm

Output players able to be traded with

Ask for input for who is to be traded with

List assets able to be traded for both parties

Ask for input determining what is being given

Ask for input determining what is being received

Ask the player selected if they accept this offer

If they accept, swap properties required and money

If they counter offer, repeat from the listing of assets

If they refuse offer, terminate function

## Jail

### IPO

#### Start of turn

|  |  |  |
| --- | --- | --- |
| ***Input*** | ***Process*** | ***Output*** |
| Roll for doubles | Call roll dice function  Check if it’s doubles | Two random numbers between 1 and 6 as ASCII dice  If doubles, out of jail |
| Manage | Call manage function | Properties owned |
| Trade | Call trade function | List of players that can trade |
| Declare bankruptcy | Call declare bankruptcy function | Player is now bankrupt |
| Terminate game | Call terminate game function | Game ending |

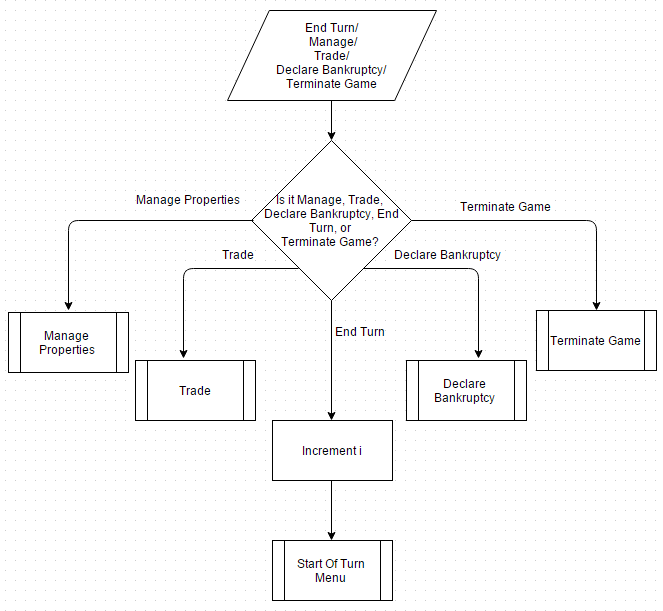
### Flowchart

#### 

#### End of turn

|  |  |  |
| --- | --- | --- |
| ***Input*** | ***Process*** | ***Output*** |
| End turn | Call stat of turn function for next i | Next players turn |
| Manage | Call manage function | Properties owned |
| Trade | Call trade function | List of players that can trade |
| Declare bankruptcy | Call declare bankruptcy function | Player is now bankrupt |
| Terminate game | Call terminate game function | Game ending |

### Flowchart



### Algorithm

#### Start of Turn

Output start of turn options along with a pay fine option

Ask for input

Determine if input was valid, if it is valid continue, otherwise repeat again

Determine action to be taken depending on input

Call appropriate function (trade, manage, declare bankruptcy, terminate game, roll dice, pay fine)

If pay fine was selected, subtract money from wallet, set player ‘inJail’ bool to false, start turn again

If roll dice was selected, call the dice roll function and determine if the roll was doubles, if it was doubles then the player is out of jail

#### End of Turn

Output end of turn options along with a pay fine option

Ask for input

Determine if input was valid, if it is valid continue, otherwise repeat again

Determine action to be taken depending on input

Call appropriate function (trade, manage, declare bankruptcy, terminate game, end turn, pay fine)

If pay fine was selected, subtract money from wallet, set player ‘inJail’ bool to false, and terminate function

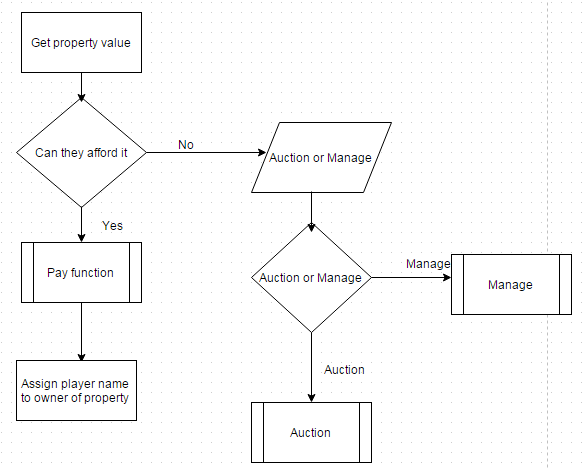
If player is still in jail, increment the turns in jail

## Purchase Property

### IPO

|  |  |  |
| --- | --- | --- |
| ***Input*** | ***Process*** | ***output*** |
| N/A | Get property value, can player afford the property.  If they can’t call auction function.  If they can, set property owner to be players name.  Subtract value from player wallet | Player purchased property. |

### Flowchart



### Algorithm

Get the property value

Determine if the player can afford to buy the property

If they can’t call the auction function

If they can, set the property owner to be the player’s name

Subtract the value from the player wallet

# Task List

## Sam

* Lead Programmer – functions include:
  + Start Turn
  + End Turn
  + Auction
  + Purchase Property
  + Building Management
  + Roll Dice
  + Trade
  + Jail
  + Main Menu
* Collaboration of documentation into a single document for final submission

## Peter

* Programming ASCII Artwork for the game
* Creator of most documentation for game modules:
  + IPO Charts
  + Flowcharts
  + Algorithms
* Meeting documentation
* Creator of Gantt charts

# Collaboration

Peter has assisted Sam with some of the programming, allowing Sam to peer-check with Peter. Sam has also assisted Peter with the creation of required documentation, providing help with the IPO charts by listing the input, processes and output for the modules, also by providing him with algorithms for the game modules. Samuel has collaborated with Peter and assisted him with the development of the IPO diagrams and the flowcharts.

# Group Meetings

## 24/06/2015

|  |  |  |
| --- | --- | --- |
| ***Present*** | ***Absent*** | ***Reason*** |
| Sam | Tom | N/A |
| Peter |  |  |

### Assessment

The groups assigned must work together to program the board game, Monopoly. It must be submitted with required documentation including progress reports, team member evaluations and other appropriate paperwork.

### Date issued

24/06/2015

### Due date

21/08/2015

### Work Conducted

We have selected our leader who’s in charge of appointing the tasks to certain members, the leader is Sam. Sam has begun writing the code and functions for the game. He is the primary code constructor. Peter has been appointed in charge of completing all the necessary documentation and writing the code for the visual effects and has begun creating the ASCII for the game. Tom didn’t attend the lesson and hasn’t attended any lessons since to the issue of the assessment and has shown no interest and participating in the development of the task.

### Encountered Problems

Tom has failed to show any interest in the task whatsoever and our group has been left with only two members to complete the task, both of the remaining two members will have to pick up extra tasks that Tom was responsible for. Sam is struggling with the coding of the arrays and as had to research and learn how to return arrays in the form of pointers.

## 26/06/2015

|  |  |  |
| --- | --- | --- |
| ***Present*** | ***Absent*** | ***Reason*** |
| Sam | Tom | N/A |
| Peter |  |  |

### Assessment

The groups assigned must work together to program the board game, Monopoly. It must be submitted with required documentation including progress reports, team member evaluations and other appropriate paperwork.

### Date issued

24/06/2015

### Due date

21/08/2015

### Work Conducted

This meeting/lesson, Sam has been busy focusing on how to write and complete the code for the jail function in monopoly. Peter has begun the documenting process of the task and has begun writing down the meetings and what work has been conducted, Peter has also began writing the ASCII code for the estates and the visual effects that will be displayed throughout the game.

### Encountered Problems

Tom has yet to show any recognition of the task, he has failed to reply to our messages in our monopoly group inbox and has failed to come to school. Some of the detailed pictures that are needed for the ASCII extremely difficult to find or replicate and a similar, less detailed ASCII picture has had to be created. The problem was that there was a semi-colon after an else if which caused the error/bug, the semi-colon has been found and erased.

## 04/07/2015

|  |  |  |
| --- | --- | --- |
| ***Present*** | ***Absent*** | ***Reason*** |
| Sam | Tom | N/A |
| Peter |  |  |

### Assessment

The groups assigned must work together to program the board game, Monopoly. It must be submitted with required documentation including progress reports, team member evaluations and other appropriate paperwork.

### Date issued

24/06/2015

### Due date

21/08/2015

### Work Conducted

Sam and Peter have both began their progress reports for the task which are due on the 17/7/2015. Peter has begun creating the Gantt chart for the task and has started creating the IPO (input, process, output) charts and the flow charts. Sam has completed the purchase and rent function for the game.

### Encountered Problems

The required documentation such as the flowcharts and IPO charts are more difficult than expected and will require some time. At approximately 2:12PM Sam’s power was cut off and half an hour of work was lost.

## 15/07/2015

|  |  |  |
| --- | --- | --- |
| ***Present*** | ***Absent*** | ***Reason*** |
| Sam | Tom | N/A |
| Peter |  |  |

### Assessment

The groups assigned must work together to program the board game, Monopoly. It must be submitted with required documentation including progress reports, team member evaluations and other appropriate paperwork.

### Date issued

24/06/2015

### Due date

21/08/2015

### Work Conducted

This session Peter has created a title page for the task, He’s also placed the ASCII estates into ‘else if’ conditions so they can be implemented into the source code with ease, Samuel has gone ahead and has implemented the ASCII and it’s working beautifully. For the rest of the session, the ASCII will be tested. The bankruptcy function has also been completed.

### Encountered Problems

There was a small bug with the ‘else if’ condition with the “GO” ASCII but has been resolved. The problem was that the function was trying to access array elements that do not exist and therefore caused the program to crash.

## 17/07/2015

|  |  |  |
| --- | --- | --- |
| ***Present*** | ***Absent*** | ***Reason*** |
| Sam |  |  |
| Peter |  |  |

### Assessment

The groups assigned must work together to program the board game, Monopoly. It must be submitted with required documentation including progress reports, team member evaluations and other appropriate paperwork.

### Date issued

24/06/2015

### Due date

21/08/2015

### Work Conducted

In this work session, Peter and Samuel both completed and printed off their progress reports and submitted them for marking Peter worked on completing all of his ASCII art for the monopoly game. Samuel has been working all day on fixing a bug in the chance cards, so far no luck.

### Encountered Problems

The bug is causing a player to pay themselves rent when they land on their own property. Peter is struggling with the ASCII as it is difficult to create detailed pictures with inaccurate symbols. Its official, Thomas has dropped the subject and is therefore, no longer part of the group. The task has been left up to Peter and Samuel

## 24/07/2015

|  |  |  |
| --- | --- | --- |
| ***Present*** | ***Absent*** | ***Reason*** |
| Sam |  |  |
| Peter |  |  |

### Assessment

The groups assigned must work together to program the board game, Monopoly. It must be submitted with required documentation including progress reports, team member evaluations and other appropriate paperwork.

### Date issued

24/06/2015

### Due date

21/08/2015

### Work Conducted

In this work session,Peter has complete all of his ASCII for the estates, stations, taxes, utilities and everything else needed for the visual effects. Samuel completed the removal of houses and hotels, implemented Peters ASCII, got rid of an unnecessary function, and overall completed the game, with a little work to polish off.

### Encountered Problems

A few errors with files, the information within the files was not correct, other than that there were no problems

## 06/08/2015

|  |  |  |
| --- | --- | --- |
| ***Present*** | ***Absent*** | ***Reason*** |
| Sam |  |  |
| Peter |  |  |

### Assessment

The groups assigned must work together to program the board game, Monopoly. It must be submitted with required documentation including progress reports, team member evaluations and other appropriate paperwork.

### Date issued

24/06/2015

### Due date

21/08/2015

### Work Conducted

Today we have completed our progress reports which are due tomorrow. Samuel has worked on completing the documents for the monopoly instructions and has polished off some things for the program. Peter has completed more ASCII for the chance and community chests.

### Encountered Problems

Peter saved the ASCII file to an unknown source and was unable to find it, therefore having to start it again. Samuel has had problems with clarifying rules.

## 18/08/2015

|  |  |  |
| --- | --- | --- |
| ***Present*** | ***Absent*** | ***Reason*** |
| Sam | Peter | N/A |

### Assessment

The groups assigned must work together to program the board game, Monopoly. It must be submitted with required documentation including progress reports, team member evaluations and other appropriate paperwork.

### Date issued

24/06/2015

### Due date

21/08/2015

### Work Conducted

Today Sam checked the code for errors and stopped the user entering invalid input. Peter failed to show up to do the flowcharts and IPO’s for the program.

### Encountered Problems

No problems today.

## 20/08/2015

|  |  |  |
| --- | --- | --- |
| ***Present*** | ***Absent*** | ***Reason*** |
| Sam |  |  |
| Peter |  |  |

### Assessment

The groups assigned must work together to program the board game, Monopoly. It must be submitted with required documentation including progress reports, team member evaluations and other appropriate paperwork.

### Date issued

24/06/2015

### Due date

21/08/2015

### Work Conducted

Sam spent the lesson commenting on his source code, Peter has begun completing his flow charts. Sam helped Peter with the flowcharts as Peter did not quite understand the algorithms to their full extent.

### Encountered Problems

No problems today.

## 21/08/2015

|  |  |  |
| --- | --- | --- |
| ***Present*** | ***Absent*** | ***Reason*** |
| Sam |  |  |
| Peter |  |  |

### Assessment

The groups assigned must work together to program the board game, Monopoly. It must be submitted with required documentation including progress reports, team member evaluations and other appropriate paperwork.

### Date issued

24/06/2015

### Due date

21/08/2015

### Work Conducted

Peter has completed all of his flowcharts and touched up the Gantt chart. Sam continued commenting on his source code and assisted Peter with the creation of the last flow charts

### Encountered Problems

No problems today.

## 24/08/2015

|  |  |  |
| --- | --- | --- |
| ***Present*** | ***Absent*** | ***Reason*** |
| Sam |  |  |
| Peter |  |  |

### Assessment

The groups assigned must work together to program the board game, Monopoly. It must be submitted with required documentation including progress reports, team member evaluations and other appropriate paperwork.

### Date issued

24/06/2015

### Due date

21/08/2015

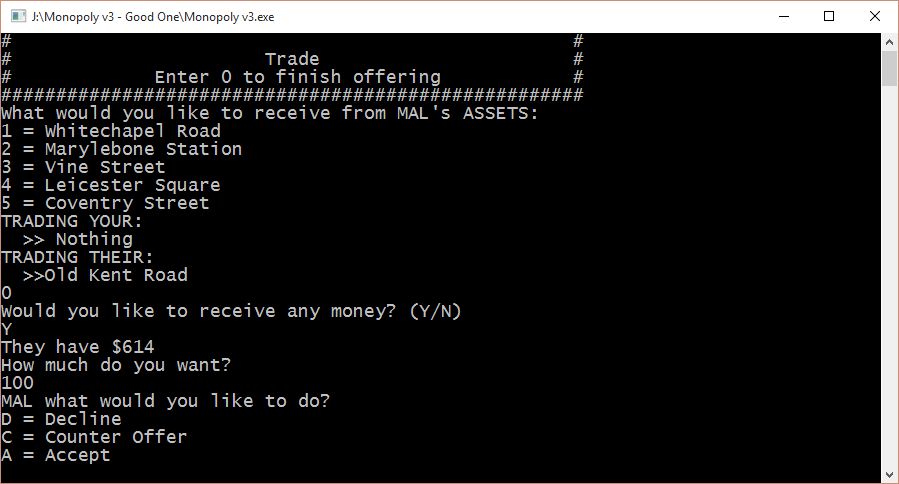
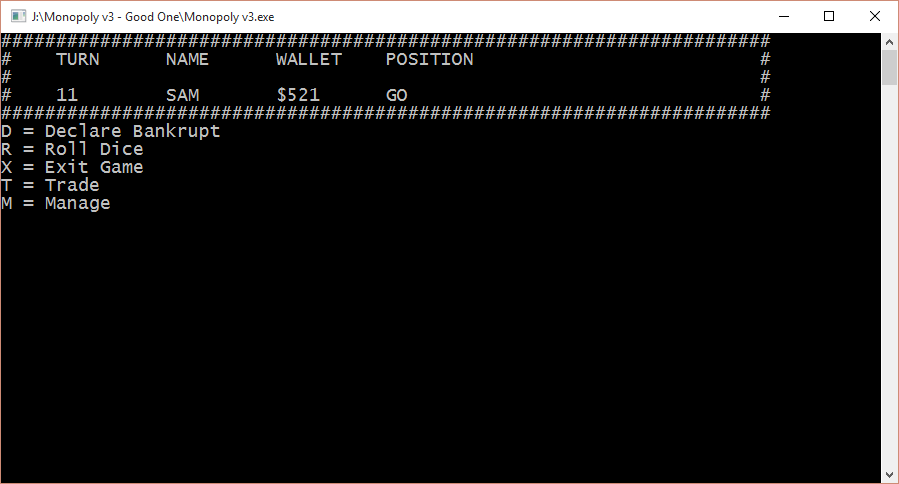
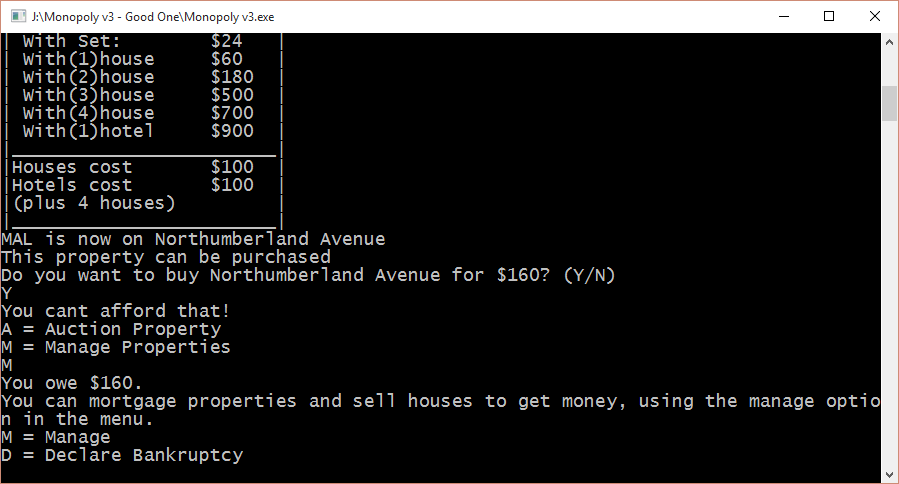
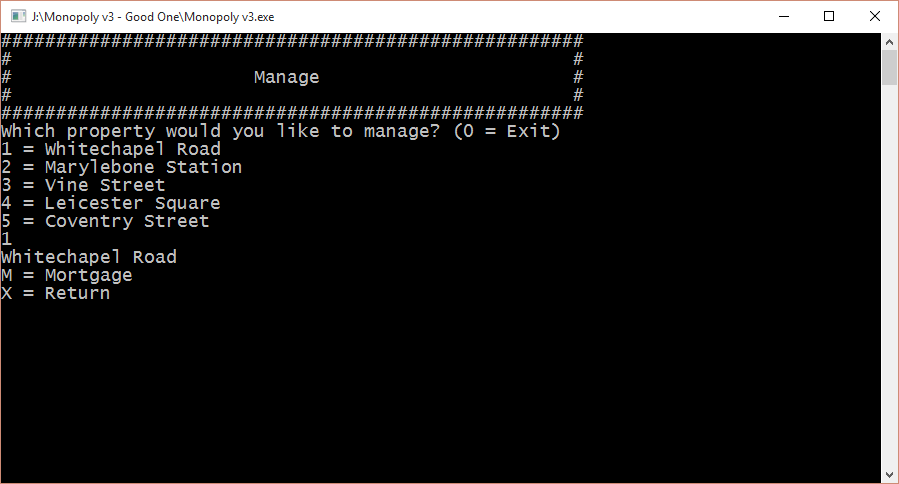
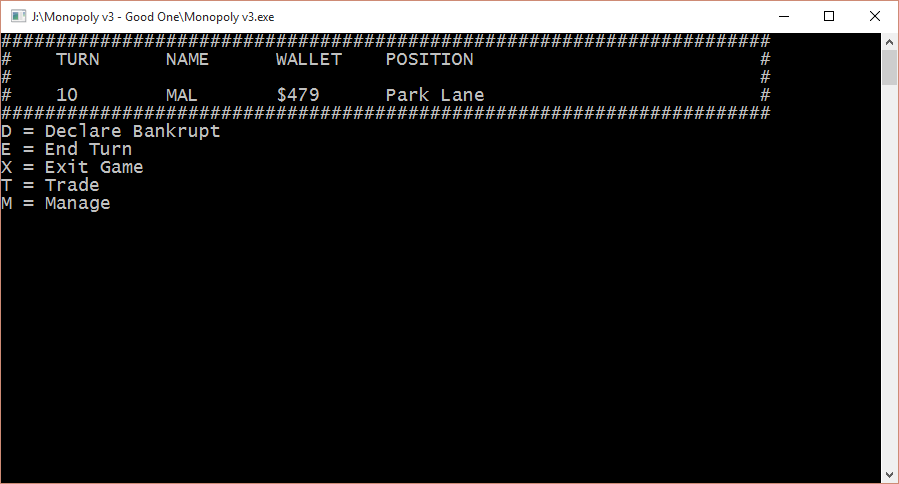
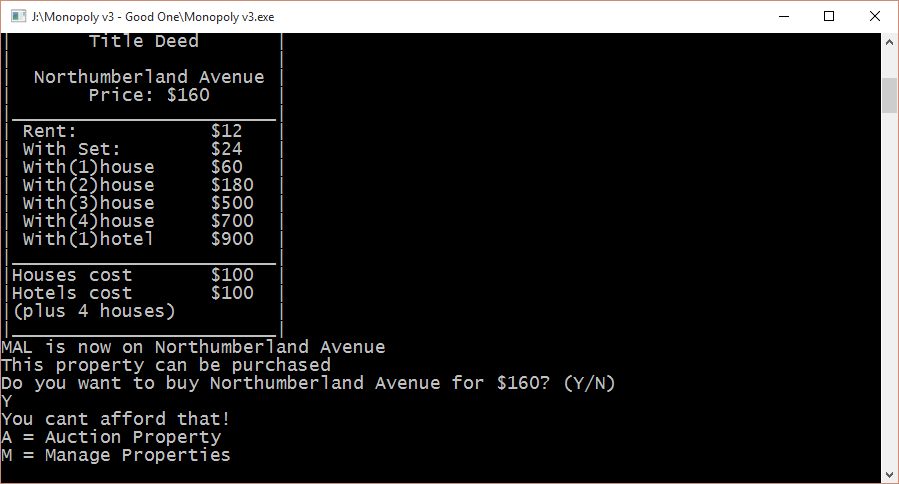
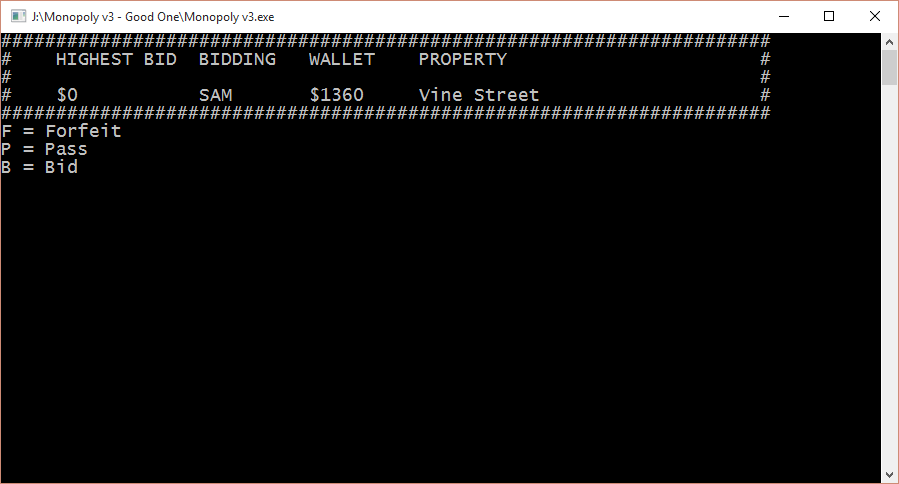
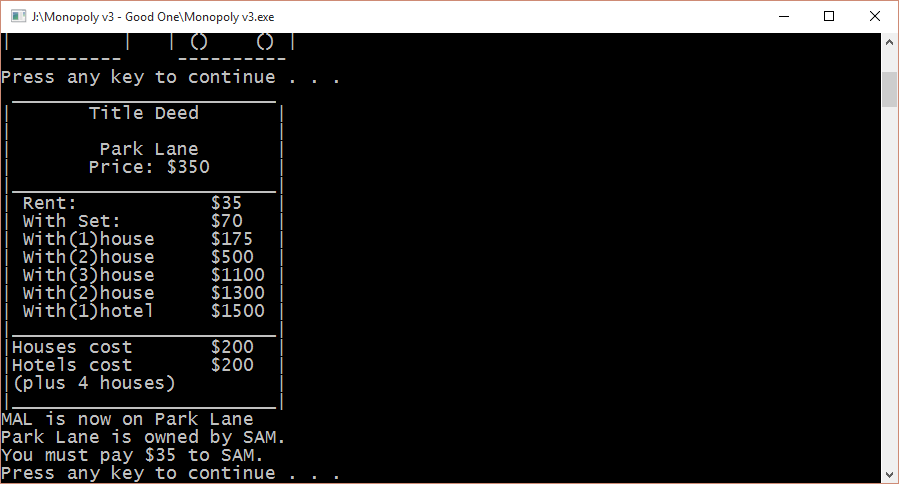
### Work Conducted

Peter modified the Gantt chart and both him and Sam completed the required documentation. They both finalised the documentation for submission.

### Encountered Problems

No problems today.

# System Testing



# Schedule

