# PA05 - Queue

Generated by Doxygen 1.8.6

Tue Nov 15 2016 15:13:06

ii CONTENTS

# **Contents**

2 File Index         2.1 File List         2           2.1 File List         2           3 Class Documentation         3           3.1 Customer Class Reference         3           3.2 Event Class Reference         2           3.3 LinkedListQueue         ItemType > Class Template Reference         4           3.3.1 Constructor & Destructor Documentation         6           3.4 Node         ItemType > Struct Template Reference         10           3.5 Queue         ItemType > Class Template Reference         10           3.5.1 Constructor & Destructor Documentation         11           3.5.2 Member Function Documentation         12           3.6.1 Constructor & Destructor Documentation         15           3.6.2 Member Function Documentation         16           3.6.3 Member Function Documentation         16           3.7.2 Member Function Documentation         16           3.7.2 Member Function Documentation         16           3.7.2 Member Function Documentation         26           3.8.1 Constructor & Destructor Documentation         26           3.8.2 Member Function Documentation         26           3.9 Simulation Class Reference         26           3.9.1 Constructor & Destructor Documentation         31           3.10 Constru	1	I Class Index			
2.1 File List       2         3 Class Documentation       3         3.1 Customer Class Reference       3         3.2 Event Class Reference       3         3.3 LinkedListQueue       Item Type > Class Template Reference         3.3.1 Constructor & Destructor Documentation       6         3.3.2 Member Function Documentation       6         3.4 Node       ItemType > Class Template Reference       10         3.5.1 Constructor & Destructor Documentation       11         3.5.2 Member Function Documentation       12         3.6.1 RadixSort Class Reference       16         3.6.1 Constructor & Destructor Documentation       17         3.6.2 Member Function Documentation       18         3.7.1 Constructor & Destructor Documentation       18         3.7.2 Member Function Documentation       20         3.8.1 Constructor & Destructor Documentation       20         3.8.2 Member Function Documentation       24         3.8.2 Member Function Documentation       25         3.9.3 Simulation3 Class Reference       25         3.9.1 Constructor & Destructor Documentation       31         3.10 Simulation-Linked1 Class Reference       32         3.10 Simulation-Linked2 Class Reference       35         3.11.1 Constructor & Destructor Document		1.1	Class List	1	
3 Class Documentation         3.1 Customer Class Reference         3.2 Event Class Reference         3.2 Event Class Reference         3.2 Event Class Reference         3.3 LinkedListQueu         4.6 Event Class Reference         3.3 LinkedListQueue         4.6 Event Class Template Reference         3.3.1 Constructor & Destructor Documentation         5.5 Queue         4.7 Event Class Template Reference         1.6 Event Class Template Reference         1.7 Event Class Template Reference         1.7 Event Class Template Reference	2	File I	• Index		
3.1       Customer Class Reference       3         3.2       Event Class Reference       3         3.3       LinkedListQueue< ItemType > Class Template Reference       4         3.3.1       Constructor & Destructor Documentation       5         3.3.2       Member Function Documentation       6         3.4       Node< ItemType > Struct Template Reference       10         3.5       Queue< ItemType > Class Template Reference       10         3.5.1       Constructor & Destructor Documentation       11         3.5.2       Member Function Documentation       12         3.6       RadixSort Class Reference       16         3.6.1       Constructor & Destructor Documentation       17         3.6.2       Member Function Documentation       18         3.7.1       Constructor & Destructor Documentation       18         3.7.2       Member Function Documentation       26         3.8.1       Constructor & Destructor Documentation       26         3.8.2       Member Function Documentation       26         3.9.1       Constructor & Destructor Documentation       30         3.10       SimulationLinked1 Class Reference       34         3.10.1       Constructor & Destructor Documentation       36     <		2.1	File List	2	
3.2       Event Class Reference       3         3.3       LinkedListQueue< ItemType > Class Template Reference       4         3.3.1       Constructor & Destructor Documentation       5         3.3.2       Member Function Documentation       6         3.4       Node       ItemType > Struct Template Reference       10         3.5       Queue       ItemType > Class Template Reference       10         3.5.1       Constructor & Destructor Documentation       11         3.5.2       Member Function Documentation       12         3.6.1       Constructor & Destructor Documentation       17         3.6.2       Member Function Documentation       18         3.7.1       Constructor & Destructor Documentation       19         3.7.2       Member Function Documentation       20         3.8.3       Simulation2 Class Reference       25         3.8.1       Constructor & Destructor Documentation       24         3.8.2       Member Function Documentation       25         3.9.1       Constructor & Destructor Documentation       30         3.9.2       Member Function Documentation       31         3.10       SimulationLinked1 Class Reference       36         3.10.1       Constructor & Destructor Do	3	Class	ass Documentation		
3.3.1       LinkedListQueue       ItemType > Class Template Reference       4         3.3.1       Constructor & Destructor Documentation       5         3.3.2       Member Function Documentation       6         3.4       Node       ItemType > Struct Template Reference       10         3.5       Queue       ItemType > Class Template Reference       10         3.5.1       Constructor & Destructor Documentation       11         3.5.2       Member Function Documentation       12         3.6.1       Constructor & Destructor Documentation       17         3.6.2       Member Function Documentation       18         3.7.1       Constructor & Destructor Documentation       18         3.7.2       Member Function Documentation       20         3.8.1       Constructor & Destructor Documentation       22         3.8.1       Constructor & Destructor Documentation       25         3.8.2       Member Function Documentation       25         3.9.1       Constructor & Destructor Documentation       31         3.10       SimulationLinked1 Class Reference       34         3.10.2       Member Function Documentation       36         3.11.1       Constructor & Destructor Documentation       36		3.1	Customer Class Reference	3	
3.3.1       Constructor & Destructor Documentation       5         3.3.2       Member Function Documentation       6         3.4       Node       ItemType > Struct Template Reference       10         3.5       Queue<       ItemType > Class Template Reference       10         3.5.1       Constructor & Destructor Documentation       11         3.5.2       Member Function Documentation       12         3.6       RadixSort Class Reference       16         3.6.1       Constructor & Destructor Documentation       17         3.6.2       Member Function Documentation       18         3.7       Simulation1 Class Reference       18         3.7.1       Constructor & Destructor Documentation       20         3.8       Simulation2 Class Reference       22         3.8.1       Constructor & Destructor Documentation       24         3.8.2       Member Function Documentation       25         3.9       Simulation3 Class Reference       25         3.9.1       Constructor & Destructor Documentation       31         3.10       SimulationLinked1 Class Reference       34         3.10.1       Constructor & Destructor Documentation       36         3.11.1       Constructor & Destructor Documentation </th <th></th> <th>3.2</th> <th>Event Class Reference</th> <th>3</th>		3.2	Event Class Reference	3	
3.3.2 Member Function Documentation       6         3.4 Node       ItemType > Struct Template Reference       10         3.5 Queue       ItemType > Class Template Reference       10         3.5.1 Constructor & Destructor Documentation       11         3.5.2 Member Function Documentation       12         3.6 RadixSort Class Reference       16         3.6.1 Constructor & Destructor Documentation       17         3.6.2 Member Function Documentation       18         3.7 Simulation1 Class Reference       18         3.7.1 Constructor & Destructor Documentation       19         3.7.2 Member Function Documentation       20         3.8 Simulation2 Class Reference       23         3.8.1 Constructor & Destructor Documentation       24         3.8.2 Member Function Documentation       25         3.9 Simulation3 Class Reference       25         3.9.1 Constructor & Destructor Documentation       30         3.9.2 Member Function Documentation       31         3.10.1 Constructor & Destructor Documentation       36         3.11.2 Member Function Documentation       36         3.11.2 Member Function Documentation       40         3.11.2 Member Function Documentation       41         3.12.1 Constructor & Destructor Documentation       45 <th></th> <th>3.3</th> <th>LinkedListQueue &lt; ItemType &gt; Class Template Reference</th> <th>4</th>		3.3	LinkedListQueue < ItemType > Class Template Reference	4	
3.4 Node       Node       ItemType > Struct Template Reference       10         3.5 Queue       ItemType > Class Template Reference       10         3.5.1 Constructor & Destructor Documentation       11         3.5.2 Member Function Documentation       12         3.6 RadixSort Class Reference       16         3.6.1 Constructor & Destructor Documentation       17         3.6.2 Member Function Documentation       18         3.7 Simulation1 Class Reference       18         3.7.1 Constructor & Destructor Documentation       19         3.7.2 Member Function Documentation       20         3.8.1 Constructor & Destructor Documentation       24         3.8.2 Member Function Documentation       25         3.9 Simulation3 Class Reference       25         3.9.1 Constructor & Destructor Documentation       36         3.9.2 Member Function Documentation       31         3.10 SimulationLinked1 Class Reference       34         3.10.1 Constructor & Destructor Documentation       36         3.11 SimulationLinked2 Class Reference       35         3.11.1 Constructor & Destructor Documentation       36         3.11.2 Member Function Documentation       40         3.11.2 Member Function Documentation       41         3.12.1 Constructor & Destructor Docume			3.3.1 Constructor & Destructor Documentation	5	
3.5. Queue       ItemType > Class Template Reference       10         3.5.1 Constructor & Destructor Documentation       11         3.5.2 Member Function Documentation       12         3.6. RadixSort Class Reference       16         3.6.1 Constructor & Destructor Documentation       17         3.6.2 Member Function Documentation       18         3.7 Simulation1 Class Reference       18         3.7.1 Constructor & Destructor Documentation       19         3.7.2 Member Function Documentation       20         3.8. Simulation2 Class Reference       23         3.8.1 Constructor & Destructor Documentation       24         3.8.2 Member Function Documentation       25         3.9 Simulation3 Class Reference       25         3.9.1 Constructor & Destructor Documentation       30         3.9.2 Member Function Documentation       31         3.10 SimulationLinked1 Class Reference       34         3.10.1 Constructor & Destructor Documentation       36         3.11 SimulationLinked2 Class Reference       35         3.11.1 Constructor & Destructor Documentation       40         3.11.2 Member Function Documentation       41         3.12.2 Member Function Documentation       46         3.12.2 Member Function Documentation       47     <			3.3.2 Member Function Documentation	6	
3.5.1       Constructor & Destructor Documentation       11         3.5.2       Member Function Documentation       12         3.6       RadixSort Class Reference       16         3.6.1       Constructor & Destructor Documentation       17         3.6.2       Member Function Documentation       18         3.7       Simulation1 Class Reference       18         3.7.1       Constructor & Destructor Documentation       19         3.7.2       Member Function Documentation       20         3.8       Simulation2 Class Reference       23         3.8.1       Constructor & Destructor Documentation       24         3.8.2       Member Function Documentation       25         3.9.1       Constructor & Destructor Documentation       30         3.9.2       Member Function Documentation       31         3.10       SimulationLinked1 Class Reference       34         3.10.1       Constructor & Destructor Documentation       36         3.11.1       Constructor & Destructor Documentation       36         3.11.2       Member Function Documentation       40         3.11.2       Member Function Documentation       41         3.12.1       Constructor & Destructor Documentation       42		3.4	Node < ItemType > Struct Template Reference	10	
3.5.2       Member Function Documentation       12         3.6       RadixSort Class Reference       16         3.6.1       Constructor & Destructor Documentation       17         3.6.2       Member Function Documentation       18         3.7       Simulation1 Class Reference       18         3.7.1       Constructor & Destructor Documentation       19         3.7.2       Member Function Documentation       20         3.8       Simulation2 Class Reference       23         3.8.1       Constructor & Destructor Documentation       24         3.8.2       Member Function Documentation       25         3.9       Simulation3 Class Reference       25         3.9.1       Constructor & Destructor Documentation       30         3.9.2       Member Function Documentation       31         3.10       SimulationLinked1 Class Reference       34         3.10.2       Member Function Documentation       36         3.11.1       Constructor & Destructor Documentation       36         3.11.2       Member Function Documentation       40         3.11.2       Member Function Documentation       41         3.12.1       Constructor & Destructor Documentation       46         3.12.2		3.5	Queue < ItemType > Class Template Reference	10	
3.6.1       Constructor & Destructor Documentation       16         3.6.1       Constructor & Destructor Documentation       17         3.6.2       Member Function Documentation       18         3.7       Simulation1 Class Reference       18         3.7.1       Constructor & Destructor Documentation       19         3.7.2       Member Function Documentation       20         3.8       Simulation2 Class Reference       23         3.8.1       Constructor & Destructor Documentation       24         3.9       Simulation3 Class Reference       25         3.9.1       Constructor & Destructor Documentation       30         3.9.2       Member Function Documentation       31         3.10       SimulationLinked1 Class Reference       34         3.10.1       Constructor & Destructor Documentation       35         3.10.2       Member Function Documentation       36         3.11.1       Constructor & Destructor Documentation       40         3.11.2       Member Function Documentation       41         3.12.2       Member Function Documentation       46         3.12.1       Constructor & Destructor Documentation       46         3.12.2       Member Function Documentation       47 <th></th> <th></th> <th>3.5.1 Constructor &amp; Destructor Documentation</th> <th>11</th>			3.5.1 Constructor & Destructor Documentation	11	
3.6.1       Constructor & Destructor Documentation       17         3.6.2       Member Function Documentation       18         3.7       Simulation1 Class Reference       18         3.7.1       Constructor & Destructor Documentation       19         3.7.2       Member Function Documentation       20         3.8       Simulation2 Class Reference       23         3.8.1       Constructor & Destructor Documentation       24         3.8.2       Member Function Documentation       25         3.9       Simulation3 Class Reference       25         3.9.1       Constructor & Destructor Documentation       30         3.9.2       Member Function Documentation       31         3.10       SimulationLinked1 Class Reference       34         3.10.1       Constructor & Destructor Documentation       36         3.11.2       Member Function Documentation       40         3.11.2       Member Function Documentation       41         3.12       SimulationLinked3 Class Reference       45         3.12.1       Constructor & Destructor Documentation       46         3.12.1       Constructor & Destructor Documentation       46         3.12.2       Member Function Documentation       47 <th></th> <th></th> <th>3.5.2 Member Function Documentation</th> <th>12</th>			3.5.2 Member Function Documentation	12	
3.6.2 Member Function Documentation       18         3.7 Simulation1 Class Reference       18         3.7.1 Constructor & Destructor Documentation       19         3.7.2 Member Function Documentation       20         3.8 Simulation2 Class Reference       25         3.8.1 Constructor & Destructor Documentation       24         3.8.2 Member Function Documentation       25         3.9 Simulation3 Class Reference       25         3.9.1 Constructor & Destructor Documentation       30         3.9.2 Member Function Documentation       31         3.10 SimulationLinked1 Class Reference       34         3.10.1 Constructor & Destructor Documentation       35         3.10.2 Member Function Documentation       36         3.11 SimulationLinked2 Class Reference       35         3.11.1 Constructor & Destructor Documentation       40         3.11.2 Member Function Documentation       41         3.12 SimulationLinked3 Class Reference       45         3.12.1 Constructor & Destructor Documentation       46         3.12.2 Member Function Documentation       47		3.6	RadixSort Class Reference	16	
3.7       Simulation1 Class Reference       18         3.7.1       Constructor & Destructor Documentation       19         3.7.2       Member Function Documentation       20         3.8       Simulation2 Class Reference       23         3.8.1       Constructor & Destructor Documentation       24         3.8.2       Member Function Documentation       25         3.9       Simulation3 Class Reference       29         3.9.1       Constructor & Destructor Documentation       30         3.9.2       Member Function Documentation       31         3.10       SimulationLinked1 Class Reference       34         3.10.1       Constructor & Destructor Documentation       36         3.10.2       Member Function Documentation       36         3.11.1       Constructor & Destructor Documentation       40         3.11.2       Member Function Documentation       41         3.12.1       Constructor & Destructor Documentation       45         3.12.1       Constructor & Destructor Documentation       46         3.12.2       Member Function Documentation       47			3.6.1 Constructor & Destructor Documentation	17	
3.7.1       Constructor & Destructor Documentation       15         3.7.2       Member Function Documentation       20         3.8       Simulation2 Class Reference       23         3.8.1       Constructor & Destructor Documentation       24         3.8.2       Member Function Documentation       25         3.9       Simulation3 Class Reference       29         3.9.1       Constructor & Destructor Documentation       30         3.9.2       Member Function Documentation       31         3.10       SimulationLinked1 Class Reference       34         3.10.1       Constructor & Destructor Documentation       36         3.10.2       Member Function Documentation       36         3.11.1       Constructor & Destructor Documentation       40         3.11.2       Member Function Documentation       41         3.12.1       Constructor & Destructor Documentation       45         3.12.1       Constructor & Destructor Documentation       46         3.12.2       Member Function Documentation       47			3.6.2 Member Function Documentation	18	
3.7.2 Member Function Documentation       20         3.8 Simulation2 Class Reference       23         3.8.1 Constructor & Destructor Documentation       24         3.8.2 Member Function Documentation       25         3.9 Simulation3 Class Reference       29         3.9.1 Constructor & Destructor Documentation       30         3.9.2 Member Function Documentation       31         3.10 SimulationLinked1 Class Reference       34         3.10.1 Constructor & Destructor Documentation       35         3.10.2 Member Function Documentation       36         3.11.1 Constructor & Destructor Documentation       40         3.11.2 Member Function Documentation       41         3.12.1 Constructor & Destructor Documentation       46         3.12.1 Constructor & Destructor Documentation       46         3.12.2 Member Function Documentation       47		3.7	Simulation1 Class Reference	18	
3.8       Simulation2 Class Reference       23         3.8.1       Constructor & Destructor Documentation       24         3.8.2       Member Function Documentation       25         3.9       Simulation3 Class Reference       25         3.9.1       Constructor & Destructor Documentation       30         3.9.2       Member Function Documentation       31         3.10       SimulationLinked1 Class Reference       34         3.10.1       Constructor & Destructor Documentation       35         3.10.2       Member Function Documentation       36         3.11.1       Constructor & Destructor Documentation       40         3.11.2       Member Function Documentation       41         3.12.1       Constructor & Destructor Documentation       46         3.12.1       Constructor & Destructor Documentation       46         3.12.2       Member Function Documentation       47			3.7.1 Constructor & Destructor Documentation	19	
3.8.1 Constructor & Destructor Documentation       24         3.8.2 Member Function Documentation       25         3.9 Simulation3 Class Reference       25         3.9.1 Constructor & Destructor Documentation       30         3.9.2 Member Function Documentation       31         3.10 SimulationLinked1 Class Reference       34         3.10.1 Constructor & Destructor Documentation       35         3.10.2 Member Function Documentation       36         3.11 SimulationLinked2 Class Reference       39         3.11.1 Constructor & Destructor Documentation       40         3.11.2 Member Function Documentation       41         3.12 SimulationLinked3 Class Reference       45         3.12.1 Constructor & Destructor Documentation       46         3.12.2 Member Function Documentation       47			3.7.2 Member Function Documentation	20	
3.8.2 Member Function Documentation       25         3.9 Simulation3 Class Reference       29         3.9.1 Constructor & Destructor Documentation       30         3.9.2 Member Function Documentation       31         3.10 SimulationLinked1 Class Reference       34         3.10.1 Constructor & Destructor Documentation       35         3.10.2 Member Function Documentation       36         3.11 SimulationLinked2 Class Reference       39         3.11.1 Constructor & Destructor Documentation       40         3.11.2 Member Function Documentation       41         3.12 SimulationLinked3 Class Reference       45         3.12.1 Constructor & Destructor Documentation       46         3.12.2 Member Function Documentation       47		3.8	Simulation2 Class Reference	23	
3.9 Simulation3 Class Reference       29         3.9.1 Constructor & Destructor Documentation       30         3.9.2 Member Function Documentation       31         3.10 SimulationLinked1 Class Reference       34         3.10.1 Constructor & Destructor Documentation       35         3.10.2 Member Function Documentation       36         3.11 SimulationLinked2 Class Reference       39         3.11.1 Constructor & Destructor Documentation       40         3.11.2 Member Function Documentation       41         3.12 SimulationLinked3 Class Reference       45         3.12.1 Constructor & Destructor Documentation       46         3.12.2 Member Function Documentation       47			3.8.1 Constructor & Destructor Documentation	24	
3.9.1 Constructor & Destructor Documentation       30         3.9.2 Member Function Documentation       31         3.10 SimulationLinked1 Class Reference       34         3.10.1 Constructor & Destructor Documentation       35         3.10.2 Member Function Documentation       36         3.11 SimulationLinked2 Class Reference       39         3.11.1 Constructor & Destructor Documentation       40         3.11.2 Member Function Documentation       41         3.12.1 Constructor & Destructor Documentation       45         3.12.1 Constructor & Destructor Documentation       46         3.12.2 Member Function Documentation       47			3.8.2 Member Function Documentation	25	
3.9.2 Member Function Documentation       31         3.10 SimulationLinked1 Class Reference       34         3.10.1 Constructor & Destructor Documentation       35         3.10.2 Member Function Documentation       36         3.11 SimulationLinked2 Class Reference       39         3.11.1 Constructor & Destructor Documentation       40         3.11.2 Member Function Documentation       41         3.12 SimulationLinked3 Class Reference       45         3.12.1 Constructor & Destructor Documentation       46         3.12.2 Member Function Documentation       47		3.9	Simulation3 Class Reference	29	
3.10 SimulationLinked1 Class Reference       34         3.10.1 Constructor & Destructor Documentation       35         3.10.2 Member Function Documentation       36         3.11 SimulationLinked2 Class Reference       39         3.11.1 Constructor & Destructor Documentation       40         3.11.2 Member Function Documentation       41         3.12 SimulationLinked3 Class Reference       45         3.12.1 Constructor & Destructor Documentation       46         3.12.2 Member Function Documentation       47			3.9.1 Constructor & Destructor Documentation	30	
3.10.1 Constructor & Destructor Documentation       35         3.10.2 Member Function Documentation       36         3.11 SimulationLinked2 Class Reference       39         3.11.1 Constructor & Destructor Documentation       40         3.11.2 Member Function Documentation       41         3.12 SimulationLinked3 Class Reference       45         3.12.1 Constructor & Destructor Documentation       46         3.12.2 Member Function Documentation       47			3.9.2 Member Function Documentation	31	
3.10.2 Member Function Documentation       36         3.11 SimulationLinked2 Class Reference       39         3.11.1 Constructor & Destructor Documentation       40         3.11.2 Member Function Documentation       41         3.12 SimulationLinked3 Class Reference       45         3.12.1 Constructor & Destructor Documentation       46         3.12.2 Member Function Documentation       47		3.10	SimulationLinked1 Class Reference	34	
3.11 SimulationLinked2 Class Reference       39         3.11.1 Constructor & Destructor Documentation       40         3.11.2 Member Function Documentation       41         3.12 SimulationLinked3 Class Reference       45         3.12.1 Constructor & Destructor Documentation       46         3.12.2 Member Function Documentation       47			3.10.1 Constructor & Destructor Documentation	35	
3.11.1 Constructor & Destructor Documentation       40         3.11.2 Member Function Documentation       41         3.12 SimulationLinked3 Class Reference       45         3.12.1 Constructor & Destructor Documentation       46         3.12.2 Member Function Documentation       47			3.10.2 Member Function Documentation	36	
3.11.2 Member Function Documentation       41         3.12 SimulationLinked3 Class Reference       45         3.12.1 Constructor & Destructor Documentation       46         3.12.2 Member Function Documentation       47		3.11	SimulationLinked2 Class Reference	39	
3.12 SimulationLinked3 Class Reference       45         3.12.1 Constructor & Destructor Documentation       46         3.12.2 Member Function Documentation       47			3.11.1 Constructor & Destructor Documentation	40	
3.12.1 Constructor & Destructor Documentation			3.11.2 Member Function Documentation	41	
3.12.2 Member Function Documentation		3.12	SimulationLinked3 Class Reference	45	
			3.12.1 Constructor & Destructor Documentation	46	
4 File Documentation			3.12.2 Member Function Documentation	47	
	4	File !	Documentation	51	

1 Class Index 1

	4.1	LinkedListQueue.cpp File Reference	51
		4.1.1 Detailed Description	51
	4.2	Queue.cpp File Reference	52
		4.2.1 Detailed Description	52
	4.3	radixSort.h File Reference	52
		4.3.1 Detailed Description	53
	4.4	Simulation1.cpp File Reference	53
		4.4.1 Detailed Description	53
	4.5	Simulation1.h File Reference	53
		4.5.1 Detailed Description	54
	4.6	Simulation2.cpp File Reference	54
		4.6.1 Detailed Description	54
	4.7	Simulation2.h File Reference	54
		4.7.1 Detailed Description	55
	4.8	Simulation3.cpp File Reference	55
		4.8.1 Detailed Description	55
	4.9	Simulation3.h File Reference	55
		4.9.1 Detailed Description	56
	4.10	SimulationLinked1.cpp File Reference	56
		4.10.1 Detailed Description	56
	4.11	SimulationLinked1.h File Reference	56
		4.11.1 Detailed Description	57
	4.12	SimulationLinked2.cpp File Reference	57
		4.12.1 Detailed Description	57
	4.13	SimulationLinked2.h File Reference	57
		4.13.1 Detailed Description	58
	4.14	SimulationLinked3.cpp File Reference	58
		4.14.1 Detailed Description	58
	4.15	SimulationLinked3.h File Reference	58
		4.15.1 Detailed Description	59
lma	dex		60
IIIC	iex		00
1	Cla	ss Index	
1.1	Cla	ass List	
He	re are	the classes, structs, unions and interfaces with brief descriptions:	
	Custo	omer	3
	Even	ıt	3

	LinkedListQueue < ItemType >	4
	Node < ItemType >	10
	Queue < ItemType >	10
	RadixSort	16
	Simulation1	18
	Simulation2	23
	Simulation3	29
	SimulationLinked1	34
	SimulationLinked2	39
	SimulationLinked3	45
2	File Index	
2.1	File List	
He	ere is a list of all documented files with brief descriptions:	
	Customer.h	??
	Event.h	??
	LinkedListQueue.cpp Implementation file for LinkedListQueue class	51
	Queue.cpp Implementation file for Queue class	52
	radixSort.h Definition file for radixSort class	52
	RadixSort.h	??
	Simulation1.cpp Implementation file for Simulation1 class	53
	Simulation1.h  Definition file for Simulation1 class	53
	Simulation2.cpp Implementation file for Simulation2 class	54
	Simulation2.h Definition file for Simulation2 class	54
	Simulation3.cpp Implementation file for Simulation3 class	55
	Simulation3.h  Definition file for Simulation3 class	55

3 Class Documentation

SimulationLinked1.cpp	
Implementation file for SimulationLinked1 class	56
SimulationLinked1.h	
Definition file for SimulationLinked1 class	56
SimulationLinked2.cpp	
Implementation file for SimulationLinked2 class	57
SimulationLinked2.h	
Definition file for SimulationLinked2 class	57
SimulationLinked3.cpp	
Implementation file for SimulationLinked3 class	58
SimulationLinked3.h	
Definition file for SimulationLinked3 class	58

# 3 Class Documentation

## 3.1 Customer Class Reference

**Public Member Functions** 

- int getArrivalTime () const
- int getWaitTime () const
- void setArrivalTime (int)
- void setWaitTime (int)
- Customer & operator= (const Customer &)

## **Private Attributes**

- int arrivalTime
- int waitTime

## Friends

ostream & operator<< (ostream &, const Customer &)</li>

The documentation for this class was generated from the following files:

- · Customer.h
- · Customer.cpp

# 3.2 Event Class Reference

**Public Member Functions** 

- Event (char, int, int)
- char **getType** () const
- int getStartTime () const
- int getLength () const
- void setStartTime (int)
- void randomizeLength (int)

## **Private Attributes**

- · char type
- · int startTime
- · int length

## Friends

ostream & operator<< (ostream &, const Event &)</li>

The documentation for this class was generated from the following files:

- · Event.h
- · Event.cpp

# 3.3 LinkedListQueue < ItemType > Class Template Reference

**Public Member Functions** 

• LinkedListQueue ()

Constructor for class Queue.

∼LinkedListQueue ()

Destructor for class Queue.

• bool isEmpty () const

Determines if queue is empty.

bool isFull () const

Determines if queue is full.

• bool clear ()

Clears the queue.

• int getLength () const

Gets length of queue.

• bool enqueue (ItemType)

Enqueues an item onto the queue.

• bool dequeue (ItemType &)

Dequeues an item from the queue.

bool peekFront (ItemType &) const

Returns front of queue.

• void print () const

Prints the queue.

• void printToFile (char \*, bool) const

Prints the queue.

# **Private Attributes**

- Node < ItemType > \* front
- Node< ItemType > \* rear

3.3.1 Constructor & Destructor Documentation		
${\it 3.3.1.1}  template < {\it class ltemType} > {\it LinkedListQueue} < {\it ltemType} > :: LinkedListQueue ( \ )$		
Constructor for class Queue.		
Able to construct a Queue object		
Precondition		
None		
Postcondition		
None		
None		
Parameters		
None		
Exceptions		
None		
Note		
: None		
$\textbf{3.3.1.2}  template < class \ ltemType > LinkedListQueue < \ ltemType > :: \sim LinkedListQueue \ (  )$		
Destructor for class Queue.		
Able to destruct a Queue object		
Precondition		
None		
Postcondition		
None		
None		
Parameters		
None		
Exceptions		
None		
Note		
: None		

3.3.2 Member Function Documentation
${\it 3.3.2.1 template}{<} {\it class ltemType} > {\it bool LinkedListQueue}{<} {\it ltemType} > {\it ::clear (} \ )$
Clears the queue.
Clears all items from the queue
Precondition
None
Postcondition
None
Algorithm
Goes through the queue and clears all items
Parameters
None
Exceptions
None
Note
: None
3.3.2.2 template < class   temType > bool LinkedListQueue <   temType > ::dequeue (     temType &   item )
Dequeues an item from the queue.
Dequeues an item from the front of the queue
Precondition
None
Postcondition
None
Algorithm
Deletes item front was holding, and sets front to next item in queue
Parameters
item
Exceptions
None
Note
: None

3.3.2.3 template < class ltemType > bool LinkedListQueue < ltemType >::enqueue ( ltemType item )
Enqueues an item onto the queue.
Enqueues an item onto the rear of the queue
Precondition
None
Postcondition
None
Algorithm
Places item after rear and sets rear to new item
Parameters
item
Exceptions
None
Note
: None
3.3.2.4 template < class ItemType > int LinkedListQueue < ItemType >::getLength ( ) const
Gets length of queue.  Returns rear - front
Precondition
None
Postcondition
None
None
Notice
Parameters
None
Exceptions
None
Note
: None

${\it 3.3.2.5}  {\it template}{<} {\it class \ ltemType} > {\it bool \ LinkedListQueue}{<} {\it ltemType} > {\it ::isEmpty ( \ \ ) \ const}$	
Determines if queue is empty.	
Determines if queue is empty, returns true if so	
Precondition	
None	
Postcondition	
None	
Algorithm	
Checks if front is equal to rear, returns true if so, returns false if it is not	
Parameters None	
None	
Exceptions	
None	
Note	
: None	
2000 tamplete calcae ham Time > heal Linked LinkOueur < ham Time > vicFull / \ ) cannot	
3.3.2.6 template < class ltemType > bool LinkedListQueue < ltemType >::isFull ( ) const	
Determines if queue is full.	
Determines if queue is full, returns true if so	
Precondition	
None	
Postcondition	
None	
Algorithm	
Checks if front is equal to rear - max, returns true if so, returns false if it is not	
Parameters None	
Exceptions	
None	
Note	
Note : None	
. NOTIG	

3.3.2.7 template < class ItemType > bool LinkedListQueue < ItemType >::peekFront ( ItemType & item ) const
Returns front of queue.
None
Precondition
None
Postcondition
None
None
Parameters
None
Exceptions
None
Note
: None
0000 tomber class hard two and birthout into the interest of the order to
3.3.2.8 template < class ltemType > void LinkedListQueue < ltemType >::print ( ) const
Prints the queue.
Prints the queue to screen
Precondition
None
Postcondition
None
Algorithm
Goes through the queue and prints out each item to screen
Parameters
None
Exceptions
None
Note
: None

The documentation for this class was generated from the following file:

• LinkedListQueue.cpp

# 3.4 Node < ItemType > Struct Template Reference

**Public Attributes** 

- ItemType data
- Node < ItemType > \* next

The documentation for this struct was generated from the following file:

• LinkedListQueue.cpp

# 3.5 Queue < ItemType > Class Template Reference

**Public Member Functions** 

• Queue (int=99999)

Constructor for class Queue.

~Queue ()

Destructor for class Queue.

• bool isEmpty () const

Determines if queue is empty.

bool isFull () const

Determines if queue is full. • int getLength () const Gets length of queue. • bool enqueue (ItemType) Enqueues an item onto the queue. • bool dequeue (ItemType &) Dequeues an item from the queue. ItemType peekFront () const Returns front of queue. • void print () const Prints the queue. void printToFile (char \*, bool) const Prints the queue. **Private Attributes** • int max · int front • int rear ItemType \* data 3.5.1 Constructor & Destructor Documentation 3.5.1.1 template < class ItemType > Queue < ItemType >::Queue ( int a = 999999 ) Constructor for class Queue. Able to construct a Queue object Precondition None Postcondition None None **Parameters** None **Exceptions** None Note

: None

3.5.1.2 template $<$ class ItemType $>$ Queue $<$ ItemType $>$ :: $\sim$ Queue ( )
Destructor for class Queue.
Able to destruct a Queue object
Precondition
None
Postcondition
None
None
Parameters
None
Exceptions
None
. Terre
Note
: None
3.5.2 Member Function Documentation
3.5.2.1 template <class ltemtype=""> bool Queue&lt; ltemType&gt;::dequeue ( ltemType &amp; item )</class>
Dequeues an item from the queue.
Dequeues an item from the front of the queue
Precondition
None
Postcondition
None
Algorithm
Deletes item front was holding, and sets front to next item in queue
Parameters
item
Exceptions
None
Note
: None

3.5.2.2 template < class	ss ItemType> bool Queue< ItemType >::enqueue ( ItemType item )			
Enqueues an item o	Enqueues an item onto the queue.			
Enqueues an item o	nto the rear of the queue			
Precondition				
None				
Postcondition				
None				
Algorithm				
Places item afte	r rear and sets rear to new item			
Parameters				
item				
Exceptions				
	None			
Note				
: None				
	ss ItemType > int Queue < ItemType >::getLength ( ) const			
Gets length of queue	<del>.</del> .			
Returns rear - front				
Precondition				
None				
Postcondition				
None				
None				
Parameters				
None				
Exceptions				
	None			
Note				
: None	: None			

${\it 3.5.2.4}  {\it template}{<} {\it class ltemType} > {\it bool Queue}{<} {\it ltemType} > {\it ::isEmpty (}  {\it ) const}$	
Determines if queue is empty.	
Determines if queue is empty, returns true if so	
Precondition	
None	
Postcondition	
None	
Algorithm	
Checks if front is equal to rear, returns true if so, returns false if it is not	
Parameters	
None	
None	
Exceptions	
None	
Note	
: None	
3.5.2.5 template < class ltemType > bool Queue < ltemType >::isFull ( ) const	
Determines if queue is full.	
Determines if queue is full, returns true if so	
Precondition	
None	
Postcondition	
None	
Algorithm	
Checks if front is equal to rear - max, returns true if so, returns false if it is not	
Parameters	
None	
Exceptions	
None	
Note	
: None	

3.5.2.6 template < class   temType   ltemType   Queue <   temType >::peekFront ( ) const
Returns front of queue.
None
Precondition
None
Nene
Postcondition
None
None
Parameters.
Parameters None
Exceptions
None
Note
: None
${\it 3.5.2.7}  {\it template}{<} {\it class \ ltemType} > {\it void \ Queue}{<} \ {\it ltemType} > {\it ::print \ (}  {\it ) \ const}$
Prints the queue.
Prints the queue to screen
Precondition
None
Postcondition
None
Algorithm
Goes through the queue and prints out each item to screen
Parameters
None
Exceptions
None
Nete
Note : None
: None

3.5.2.8 template < class ItemType > void Queue < ItemType >::printToFile ( char \* fileName, bool append ) const

Prints the queue.

Prints the queue to a file

Precondition
None

Postcondition
None

Algorithm
Goes through the queue and prints out each item to a file

Parameters

None

Exceptions

None

The documentation for this class was generated from the following file:

Queue.cpp

: None

# 3.6 RadixSort Class Reference

**Public Member Functions** 

• RadixSort ()

Constructor for class RadixSort.

∼RadixSort ()

Destructor for class RadixSort.

• void sort (int \*data, int size, int digits)

Sorts data from least to greatest.

- int getComparisonNum ()
- int getSwapNum ()
- void sort (int \*data, int size, int digits)
- int getComparisonNum ()
- int getSwapNum ()

## **Private Attributes**

- int numberOfComparisons
- · int numberOfSwaps

3.6.1 Constructor & Destructor Documentation
3.6.1.1 RadixSort::RadixSort ( )
Constructor for class RadixSort.
Able to construct a RadixSort object
Precondition
None
Postcondition
None
None
Parameters
None
None
Exceptions
None
Note
: None
3.6.1.2 RadixSort::∼RadixSort()
Destructor for class RadixSort.
Able to destruct a RadixSort object
Descendition
Precondition
None
Postcondition
None
None
Parameters
None
Eventions
Exceptions None
None
Note
: None

#### 3.6.2 Member Function Documentation

3.6.2.1 void RadixSort::sort ( int \* data, int size, int digits )

Sorts data from least to greatest.

Sorts data by grouping it together by the data's speicific digits

#### Precondition

numberOfComparisons begins at 1 numberOfSwaps begins at 1

#### Postcondition

numberOfComparisons gets incremented every time function compares values numberOfComparisons gets incremented every time function performs an action that moves the data around

## Algorithm

Radix sort is a complex algorithm that groups the data together first by the least significant digit and keeps going all the way until it reaches the most significant digit. When it groups them together, it puts them in groups of 0s - 9s, all depending on what the digit is in that specific place. It then replaces them back into the array and moves on to the next least significant bit. By the time it is done with the most significant bit, all of the items are sorted.

## **Parameters**

data	This is the data the function will sort
size	The size of data
digits	The max amount of digits there can be in a single item

# **Exceptions**

None	

#### Note

: None

The documentation for this class was generated from the following files:

- · radixSort.h
- · RadixSort.cpp

#### 3.7 Simulation1 Class Reference

**Public Member Functions** 

• Simulation1 (int=99999)

Constructor for class Simulation1.

∼Simulation1 ()

Destructor for class Simulation1.

void getArrivals (char \*)

Gets input from file.

· void simulate ()

Simulates a bank.

void processArrival (Queue < Event > &, Queue < Event > &, Queue < Event > &, Queue < int > &)

processes an arrival

void processDeparture (Queue < Event > &, Queue < Event > &, Queue < Event > &, Queue < int > &)
 processes a departure

## **Private Attributes**

- · bool firstSim
- Queue < Customer > customerQueue
- Event customer
- Event \* arrivals
- int numOfCustomers
- int numOfEvents
- int currentTime
- int transactionTime
- int maxWaitTime
- int totalWaitTime
- int maxLengthOfLine
- float averageWaitTime
- float averageLengthOfLine
- · float totalLengthOfLine
- bool tellerAvailable
- int totalldleTime
- · int idleStart
- int idleEnd

3.7.1	Constructor	& Destructor	<b>Documentation</b>
-------	-------------	--------------	----------------------

3.7.1.1 Simulation1::Simulation1 ( int a = 999999 )

Constructor for class Simulation1.

Able to construct a Simulation1 object

Precondition

None

Postcondition

None

None

Para	me	ters
------	----	------

а	defaults at 99999
---	-------------------

# **Exceptions**

None
------

Note

: None

3.7.1.2 Simulation1::~Simulation1 ( )
Destructor for class Simulation1.
Able to destruct a Simulation1 object
Precondition
None
Postcondition
None
None
Parameters
None
Exceptions
None
None
Note
: None
3.7.2 Member Function Documentation
3.7.2.1 void Simulation1::getArrivals ( char * fileName )
Gets input from file.
Gets input from file, specifically, a file of integers in numerical order that will be used for the arrivalQueue
Precondition
arrivals is empty
arrivals is empty
Postcondition
arrivals contains all of the integers from the file
Algorithm
For the amount of customers in this simulation, loop goes through the specified file and sets the parameters of the array of events called arrivals
Parameters
fileName
Exceptions

None

Note

: None

3.7.2.2 void Simulation1::processArrival ( Queue < Event > & arrivalQueue, Queue < Event > & eventQueue, Queue < Event > & bankLine, Queue < int > & departureQueue )

processes an arrival

dequeus the arrivalQueue, enqueues the eventQueue, enqueues the bankLine if necessary otherwise it enqueues the departureQueue

#### Precondition

tellerAvailable will either be available or unavailable

#### Postcondition

tellerAvailable will be unavailable if it was available

# **Algorithm**

First, it dequeues from the arrivalQueue and then enqueues that on to the eventQueue. Then it creates a customer and sets its arrival time, which is necessary for tracking its wait time. If the line is empty and a teller is available, then wait time is 0, determines the departure time, enqueues that on to departureQueue, ends the idle time for the teller, and sets tellerAvailable to false. Otherwise it enqueues the customer's arrival onto the correct bankLine.

# **Parameters**

	see	details for parameter specifications
Exceptions		
		None

Note

: None

3.7.2.3 void Simulation1::processDeparture ( Queue < Event > & arrivalQueue, Queue < Event > & eventQueue, Queue < Event > & bankLine, Queue < int > & departureQueue )

processes a departure

dequeus the departure, enqueues the eventQueue, dequeues the bankLine if necessary

Precondition

tellerAvailable will either be unavailable

## Postcondition

tellerAvailable will be available if the bankLine is empty

## Algorithm

First, it dequeues from the departureQueue and then enqueus that on to the eventQueue. It checks if the correct bankLine is empty. If it is not, then it dequeus the customer from that bank line and determines the customer's wait time. It is then able to determine the departure time of that customer. If the correct bankLine is not empty anymore then the teller's idleTime starts and its availability is set to true.

#### **Parameters**

see details for parameter specifications

#### **Exceptions**

None

Note

: None

3.7.2.4 void Simulation1::simulate ( )

Simulates a bank.

Simulates a bank with one teller and one line

Precondition

eventQueue, cutsomerQueue, bankLine, and departureQueue are empty arrivalQueue is immediately filled with arrivals

## Postcondition

As arrivalQueue is dequeued, the other queues get enqueued with data and then dequeued according to the arrivals' specification

## **Algorithm**

My implementation is a little bit different than the book's. Instead of a priority queue, I used three different queues, and arrivalQueue, departureQueue, and an eventQueue. The eventQueue does not do anything except store all the events that occur throughout the algorithm for output at the end of the algorithm. arrivalQueue is instantly enqueued with all of its data and it no longer gets enqueued throughout the algorithm. After enqueing all of the arrivals, it checks what is at the front of the arrivalQueue and at the front of departureQueue. It then uses this data to determine which one it will process. If the arrivalQueue's front time is earlier than the departureQueue's, then it will process an arrival first and vice versa. It also checks if the departureQueue is empty, which it is at the very beginning. It then processes an arrival if this is so, which is always true at the beginning. A fourth queue, bankLine, is used to store what customers are still waiting in line by simply enqueing the arrival event when it occurs and then dequeing it once the departure time of the event has been determined. A fifth queue, customerQueue, is only used to store the wait time of each arriving customer. After the first loop finishes, there are still departures that haven't been processed, so it empties the bank line and processes the departures.

# **Parameters**

None

## **Exceptions**

None

Note

: None

The documentation for this class was generated from the following files:

- · Simulation1.h
- Simulation1.cpp

# 3.8 Simulation2 Class Reference

#### **Public Member Functions**

• Simulation2 (int=99999)

Constructor for class Simulation2.

∼Simulation2 ()

Destructor for class Simulation2.

void getArrivals (char \*)

Gets input from file.

• int getShortestLine ()

Gets the shortest line.

• int getCurrentLine (int)

Gets the departuring line.

void getTellerAvailability (int, int)

Sets each teller's availability.

· void simulate ()

Simulates a bank.

- void processArrival (Queue < Event > &, Queue < Event > &, Queue < Event > &, Queue < int > &)
   processes an arrival
- void processDeparture (Queue < Event > &, Queue < Event > &, Queue < Event > &, Queue < int > &)
   processes a departure

#### **Private Attributes**

- · bool firstSim
- Queue < Customer > customerQueue
- Queue < Event > bankLine1
- Queue < Event > bankLine2
- Queue < Event > bankLine3
- Event customer
- Event \* arrivals
- int numOfCustomers
- int numOfEvents
- int currentTime
- int transactionTime
- int maxWaitTime
- int totalWaitTime
- int maxLengthOfLine1
- int maxLengthOfLine2
- int maxLengthOfLine3
- float averageWaitTime
- float averageLengthOfLine1
- · float totalLengthOfLine1
- float averageLengthOfLine2
- float totalLengthOfLine2
- float averageLengthOfLine3
- float totalLengthOfLine3
- bool tellerAvailable
- bool teller1
- · bool teller2
- · bool teller3
- bool emptyLine

<ul> <li>bool emptyLineSpec</li> <li>int totalldleTime1</li> <li>int idleStart1</li> <li>int idleEnd1</li> <li>int totalldleTime2</li> <li>int idleStart2</li> <li>int idleEnd2</li> <li>int totalldleTime3</li> <li>int idleStart3</li> <li>int idleEnd3</li> </ul>
3.8.1 Constructor & Destructor Documentation
<b>3.8.1.1 Simulation2::Simulation2 (int </b> <i>a</i> = 999999 <b>)</b>
Constructor for class Simulation2.
Able to construct a Simulation2 object
Precondition
None
Postcondition
None
None
Parameters  a defaults at 99999
Exceptions
None
Note
: None
3.8.1.2 Simulation2::∼Simulation2()
Destructor for class Simulation2.
Able to destruct a Simulation2 object
Precondition
None
Postcondition
None
None

Parameters
None
Exceptions
None
Note
: None
3.8.2 Member Function Documentation
3.8.2.1 void Simulation2::getArrivals ( char * fileName )
Gets input from file.
Gets input from file, specifically, a file of integers in numerical order that will be used for the arrivalQueue
Precondition
arrivals is empty
Postcondition
arrivals contains all of the integers from the file
Algorithm
For the amount of customers in this simulation, loop goes through the specified file and sets the parameters of the array of events called arrivals
Parameters
fileName
Exceptions
None
Note
: None
. Notice
3.8.2.2 int Simulation2::getCurrentLine ( int time )
Gets the departuring line.
Gets the line number for the correlating departing line
Precondition
None
Postcondition
None
Algorithm
Compares time to the front event's start time for each line and returns the correlating line number

Parameters		
time		
Exceptions		
	None	
Note		
: None		
3.8.2.3 int Simulation	e::getShortestLine ( )	
Gets the shortest line	<del>)</del> .	
Gets the shortest lin	out of three lines	
Precondition		
shortestLine de	efaults at 1	
Postcondition		
if any line leng	h is less than 1, shortestLine gets that line number	
Algorithm		
see precondition	and postcondition also sets emptyLine to true if either of the three lines are empty	
Parameters		
None		
Everations		
Exceptions	None	
	None	
Note		
: None		
3.8.2.4 void Simulation	n2::getTellerAvailability ( int tellerNumber, int time )	
Sets each teller's ava	uilability.	
	railability and sets the general tellerAvailable boolean based on the three tellers' indivi- nd ends idle time for each.	dua
Precondition		
teller1, 2, and	B, as well as tellerAvailability	
Postcondition		
preconditions (	et set based on parameters	
Algorithm		

Depending on the tellerNumber, sets correlating teller's data based on time;

#### **Parameters**

teller-	
Number,time	

## **Exceptions**

None	

Note

: None

3.8.2.5 void Simulation2::processArrival ( Queue < Event > & arrivalQueue, Queue < Event > & eventQueue, Queue < Event > & bankLine, Queue < int > & departureQueue )

processes an arrival

dequeus the arrivalQueue, enqueues the eventQueue, enqueues the bankLine if necessary otherwise it enqueues the departureQueue

## Precondition

tellerAvailable will either be available or unavailable

#### Postcondition

tellerAvailable will be unavailable if it was available

## Algorithm

First, it dequeues from the arrivalQueue and then enqueues that on to the eventQueue. Then it creates a customer and sets its arrival time, which is necessary for tracking its wait time. If the line is empty and a teller is available, then wait time is 0, determines the departure time, enqueues that on to departureQueue, ends the idle time for the teller, and sets tellerAvailable to false. Otherwise it enqueues the customer's arrival onto the correct bankLine.

# **Parameters**

see	details for parameter specifications	
Exceptions		
	None	

Note

: None

3.8.2.6 void Simulation2::processDeparture ( Queue < Event > & arrivalQueue, Queue < Event > & eventQueue, Queue < Event > & bankLine, Queue < int > & departureQueue )

processes a departure

dequeus the departure, enqueues the eventQueue, dequeues the bankLine if necessary

## Precondition

tellerAvailable will either be unavailable

#### Postcondition

tellerAvailable will be available if the bankLine is empty

#### Algorithm

First, it dequeues from the departureQueue and then enqueus that on to the eventQueue. It checks if the correct bankLine is empty. If it is not, then it dequeus the customer from that bank line and determines the customer's wait time. It is then able to determine the departure time of that customer. If the correct bankLine is not empty anymore then the teller's idleTime starts and its availability is set to true.

#### **Parameters**

see details for parameter specifications				
Exceptions				
	None			

#### Note

: None

3.8.2.7 void Simulation2::simulate ( )

Simulates a bank.

Simulates a bank with three tellers and three lines

#### Precondition

eventQueue, cutsomerQueue, bankLine (3), and departureQueue are empty arrivalQueue is immediately filled with arrivals

#### Postcondition

As arrivalQueue is dequeued, the other queues get enqueued with data and then dequeued according to the arrivals' specification

## Algorithm

My implementation is a little bit different than the book's. Instead of a priority queue, I used three different queues, and arrivalQueue, departureQueue, and an eventQueue. The eventQueue does not do anything except store all the events that occur throughout the algorithm for output at the end of the algorithm. arrivalQueue is instantly enqueued with all of its data and it no longer gets enqueued throughout the algorithm. After enqueing all of the arrivals, it checks what is at the front of the arrivalQueue and at the front of departureQueue. It then uses this data to determine which one it will process. If the arrivalQueue's front time is earlier than the departureQueue's, then it will process an arrival first and vice versa. It also checks if the departureQueue is empty, which it is at the very beginning. It then processes an arrival if this is so, which is always true at the beginning. The multiple bankLine queues, are used to store what customers are still waiting in line by simply enqueing the arrival event when it occurs and then dequeing it once the departure time of the event has been determined. A fifth queue, customerQueue, is only used to store the wait time of each arriving customer. After the first loop finishes, there are still departures that haven't been processed, so it empties the bank line and processes the departures.

#### **Parameters**

None	

# **Exceptions**

None	

#### Note

: Departure upon output will be out of order simply because I did not have enough time to implement more departureQueues to correlate with the multiple tellers. This means that the departureQueue is just out of order because its holding what is technically three different queues. The algorithm is run correctly though and the data is mostly correct as well.

The documentation for this class was generated from the following files:

- · Simulation2.h
- · Simulation2.cpp

# 3.9 Simulation3 Class Reference

#### **Public Member Functions**

• Simulation3 (int=99999)

Constructor for class Simulation3.

∼Simulation3 ()

Destructor for class Simulation3.

void getArrivals (char \*)

Gets input from file.

• int getTellerAvailability ()

Determines teller's availability.

• int getTeller ()

Returns which teller is unavailable.

• void simulate ()

Simulates a bank.

- void processArrival (Queue < Event > &, Queue < Event > &, Queue < Event > &, Queue < int > &)
   processes an arrival
- void processDeparture (Queue < Event > &, Queue < Event > &, Queue < Event > &, Queue < int > &)
   processes a departure

## **Private Attributes**

- · bool firstSim
- Queue < Customer > customerQueue
- Event customer
- Event \* arrivals
- int numOfCustomers
- int numOfEvents
- int currentTime
- int transactionTime
- int maxWaitTime
- int totalWaitTime
- int maxLengthOfLine

•	float averageWaitTime
•	float averageLengthOfLine
•	float totalLengthOfLine
•	bool tellerAvailable

- · bool teller1
- bool teller2
- bool teller2bool teller3
- int totalIdleTime1
- int idleStart1
- int idleEnd1
- int totalIdleTime2
- int idleStart2
- int idleEnd2
- int totalIdleTime3
- int idleStart3
- int idleEnd3

3.9	1 (	Constructor	& Destructor	Documentation

3.9.1.1 Simulation3::Simulation3 ( int a = 999999 )

Constructor for class Simulation3.

Able to construct a Simulation3 object

Precondition

None

Postcondition

None

None

**Parameters** 

a defaults at 99999

**Exceptions** 

None

Note

: None

3.9.1.2 Simulation3::~Simulation3 ( )

Destructor for class Simulation3.

Able to destruct a Simulation3 object

Precondition

None

Postcondition
None
None
Parameters
None
Exceptions
None
Note
: None
3.9.2 Member Function Documentation
3.9.2.1 void Simulation3::getArrivals ( char * fileName )
Gets input from file.
Gets input from file, specifically, a file of integers in numerical order that will be used for the arrivalQueue
Precondition
arrivals is empty
Postcondition
arrivals contains all of the integers from the file
Algorithm
For the amount of customers in this simulation, loop goes through the specified file and sets the parameters of the array of events called arrivals
Parameters  fileName
Exceptions None
None
Note
: None
3.9.2.2 int Simulation3::getTeller ( )
Returns which teller is unavailable.
None
Precondition
None

32 **CONTENTS** Postcondition None Algorithm If teller is unavailable, return teller number **Parameters** None **Exceptions** None Note : None 3.9.2.3 int Simulation3::getTellerAvailability ( ) Determines teller's availability. Returns which teller is available Precondition None Postcondition None Algorithm Depending on the data of the individual tellers, returns specific teller's number if it is available **Parameters** None **Exceptions** None Note : None 3.9.2.4 void Simulation3::processArrival ( Queue < Event > & arrivalQueue, Queue < Event > & eventQueue, Queue < **Event** > & bankLine, Queue < int > & departureQueue ) processes an arrival dequeus the arrivalQueue, enqueues the eventQueue, enqueues the bankLine if necessary otherwise it enqueues the departureQueue Precondition tellerAvailable will either be available or unavailable

## Postcondition

tellerAvailable will be unavailable if it was available

## Algorithm

First, it dequeues from the arrivalQueue and then enqueues that on to the eventQueue. Then it creates a customer and sets its arrival time, which is necessary for tracking its wait time. If the line is empty and a teller is available, then wait time is 0, determines the departure time, enqueues that on to departureQueue, ends the idle time for the teller, and sets tellerAvailable to false. Otherwise it enqueues the customer's arrival onto the correct bankLine.

#### **Parameters**

	see	details for parameter specifications
Exceptions		
		None

#### Note

: None

3.9.2.5 void Simulation3::processDeparture ( Queue < Event > & arrivalQueue, Queue < Event > & eventQueue, Queue < Event > & bankLine, Queue < int > & departureQueue )

processes a departure

dequeus the departure, enqueues the eventQueue, dequeues the bankLine if necessary

## Precondition

tellerAvailable will either be unavailable

#### Postcondition

tellerAvailable will be available if the bankLine is empty

## **Algorithm**

First, it dequeues from the departureQueue and then enqueus that on to the eventQueue. It checks if the correct bankLine is empty. If it is not, then it dequeus the customer from that bank line and determines the customer's wait time. It is then able to determine the departure time of that customer. If the correct bankLine is not empty anymore then the teller's idleTime starts and its availability is set to true.

## **Parameters**

S	ee	details for parameter specifications	
Exceptions			
		None	

## Note

: None

3.9.2.6 void Simulation3::simulate ( )

Simulates a bank.

Simulates a bank with three tellers and one lines

#### Precondition

eventQueue, cutsomerQueue, bankLine, and departureQueue are empty arrivalQueue is immediately filled with arrivals

#### Postcondition

As arrivalQueue is dequeued, the other queues get enqueued with data and then dequeued according to the arrivals' specification

## Algorithm

My implementation is a little bit different than the book's. Instead of a priority queue, I used three different queues, and arrivalQueue, departureQueue, and an eventQueue. The eventQueue does not do anything except store all the events that occur throughout the algorithm for output at the end of the algorithm. arrivalQueue is instantly enqueued with all of its data and it no longer gets enqueued throughout the algorithm. After enqueing all of the arrivals, it checks what is at the front of the arrivalQueue and at the front of departureQueue. It then uses this data to determine which one it will process. If the arrivalQueue's front time is earlier than the departureQueue's, then it will process an arrival first and vice versa. It also checks if the departureQueue is empty, which it is at the very beginning. It then processes an arrival if this is so, which is always true at the beginning. The bankLine queues is used to store what customers are still waiting in line by simply enqueing the arrival event when it occurs and then dequeing it once the departure time of the event has been determined. A fifth queue, customerQueue, is only used to store the wait time of each arriving customer. After the first loop finishes, there are still departures that haven't been processed, so it empties the bank line and processes the departures.

#### **Parameters**

None		
Exceptions		
	None	

#### Note

: Departure upon output will be out of order simply because I did not have enough time to implement more departureQueues to correlate with the multiple tellers. This means that the departureQueue is just out of order because its holding what is technically three different queues. The algorithm is run correctly though and the data is mostly correct as well.

The documentation for this class was generated from the following files:

- · Simulation3.h
- · Simulation3.cpp

# 3.10 SimulationLinked1 Class Reference

## **Public Member Functions**

SimulationLinked1 (int=99999)

Constructor for class SimulationLinked1.

∼SimulationLinked1 ()

Destructor for class SimulationLinked1.

void getArrivals (char \*)

Gets input from file.

· void simulate ()

Simulates a bank.

void processArrival (LinkedListQueue < Event > &, LinkedListQueue < Event > &, LinkedListQueue < Event > &, LinkedListQueue < int > &)

processes an arrival

void processDeparture (LinkedListQueue< Event > &, LinkedListQueue< Event > &, LinkedListQueue<</li>
 Event > &, LinkedListQueue<</li>

processes a departure

### **Private Attributes**

- · bool firstSim
- LinkedListQueue < Customer > customerQueue
- Event customer
- Event \* arrivals
- · int numOfCustomers
- int numOfEvents
- int currentTime
- · int transactionTime
- int maxWaitTime
- · int totalWaitTime
- · int maxLengthOfLine
- float averageWaitTime
- float averageLengthOfLine
- float totalLengthOfLine
- bool tellerAvailable
- int totalldleTime
- int idleStart
- int idleEnd

# 3.10.1 Constructor & Destructor Documentation

3.10.1.1 SimulationLinked1::SimulationLinked1 (int a = 999999)

Constructor for class SimulationLinked1.

Able to construct a SimulationLinked1 object

Precondition

None

Postcondition

None

None

**Parameters** defaults at 99999 **Exceptions** None Note : None 3.10.1.2 SimulationLinked1::~SimulationLinked1 ( ) Destructor for class SimulationLinked1. Able to destruct a SimulationLinked1 object Precondition None Postcondition None None **Parameters** None **Exceptions** None Note : None 3.10.2 Member Function Documentation 3.10.2.1 void SimulationLinked1::getArrivals ( char \* fileName ) Gets input from file. Gets input from file, specifically, a file of integers in numerical order that will be used for the arrivalQueue Precondition arrivals is empty Postcondition

arrivals contains all of the integers from the file

# **Algorithm**

For the amount of customers in this simulation, loop goes through the specified file and sets the parameters of the array of events called arrivals

**Parameters** 

fileName

**Exceptions** 

None

Note

: None

3.10.2.2 void SimulationLinked1::processArrival ( LinkedListQueue < Event > & arrivalQueue, LinkedListQueue < Event > & eventQueue, LinkedListQueue < Event > & bankLine, LinkedListQueue < int > & departureQueue )

processes an arrival

dequeus the arrivalQueue, enqueues the eventQueue, enqueues the bankLine if necessary otherwise it enqueues the departureQueue

Precondition

tellerAvailable will either be available or unavailable

Postcondition

tellerAvailable will be unavailable if it was available

### Algorithm

First, it dequeues from the arrivalQueue and then enqueues that on to the eventQueue. Then it creates a customer and sets its arrival time, which is necessary for tracking its wait time. If the line is empty and a teller is available, then wait time is 0, determines the departure time, enqueues that on to departureQueue, ends the idle time for the teller, and sets tellerAvailable to false. Otherwise it enqueues the customer's arrival onto the correct bankLine.

**Parameters** 

see	details for parameter specifications

**Exceptions** 

None

Note

: None

3.10.2.3 void SimulationLinked1::processDeparture ( LinkedListQueue< Event > & arrivalQueue, LinkedListQueue< Event > & bankLine, LinkedListQueue< int > & departureQueue )

processes a departure

dequeus the departure, enqueues the eventQueue, dequeues the bankLine if necessary

Precondition

tellerAvailable will either be unavailable

#### Postcondition

tellerAvailable will be available if the bankLine is empty

### Algorithm

First, it dequeues from the departureQueue and then enqueus that on to the eventQueue. It checks if the correct bankLine is empty. If it is not, then it dequeus the customer from that bank line and determines the customer's wait time. It is then able to determine the departure time of that customer. If the correct bankLine is not empty anymore then the teller's idleTime starts and its availability is set to true.

#### **Parameters**

	see	details for parameter specifications
Exceptions		
		None

#### Note

: None

3.10.2.4 void SimulationLinked1::simulate ( )

Simulates a bank.

Simulates a bank with one teller and one line

#### Precondition

eventQueue, cutsomerQueue, bankLine, and departureQueue are empty arrivalQueue is immediately filled with arrivals

### Postcondition

As arrivalQueue is dequeued, the other queues get enqueued with data and then dequeued according to the arrivals' specification

#### Algorithm

My implementation is a little bit different than the book's. Instead of a priority queue, I used three different queues, and arrivalQueue, departureQueue, and an eventQueue. The eventQueue does not do anything except store all the events that occur throughout the algorithm for output at the end of the algorithm. arrivalQueue is instantly enqueued with all of its data and it no longer gets enqueued throughout the algorithm. After enqueing all of the arrivals, it checks what is at the front of the arrivalQueue and at the front of departureQueue. It then uses this data to determine which one it will process. If the arrivalQueue's front time is earlier than the departureQueue's, then it will process an arrival first and vice versa. It also checks if the departureQueue is empty, which it is at the very beginning. It then processes an arrival if this is so, which is always true at the beginning. A fourth queue, bankLine, is used to store what customers are still waiting in line by simply enqueing the arrival event when it occurs and then dequeing it once the departure time of the event has been determined. A fifth queue, customerQueue, is only used to store the wait time of each arriving customer. After the first loop finishes, there are still departures that haven't been processed, so it empties the bank line and processes the departures.

#### **Parameters**

None	

# **Exceptions**

None	

### Note

: None

The documentation for this class was generated from the following files:

- · SimulationLinked1.h
- SimulationLinked1.cpp

### 3.11 SimulationLinked2 Class Reference

### **Public Member Functions**

• SimulationLinked2 (int=99999)

Constructor for class SimulationLinked2.

∼SimulationLinked2 ()

Destructor for class SimulationLinked2.

void getArrivals (char \*)

Gets input from file.

• int getShortestLine ()

Gets the shortest line.

int getCurrentLine (int)

Gets the departuring line.

void getTellerAvailability (int, int)

Sets each teller's availability.

• void simulate ()

Simulates a bank.

void processArrival (LinkedListQueue < Event > &, LinkedListQueue < Event > &, LinkedListQueue < Event > &, LinkedListQueue < int > &)

processes an arrival

void processDeparture (LinkedListQueue< Event > &, LinkedListQueue< Event > &, LinkedListQueue<</li>
 Event > &, LinkedListQueue<</li>

processes a departure

### **Private Attributes**

- · bool firstSim
- LinkedListQueue < Customer > customerQueue
- LinkedListQueue < Event > bankLine1
- LinkedListQueue < Event > bankLine2
- LinkedListQueue < Event > bankLine3
- Event customer
- Event \* arrivals
- int numOfCustomers
- int numOfEvents
- int currentTime

- int transactionTime
- int maxWaitTime
- · int totalWaitTime
- · int maxLengthOfLine1
- int maxLengthOfLine2
- int maxLengthOfLine3
- float averageWaitTime
- float averageLengthOfLine1
- float totalLengthOfLine1
- float averageLengthOfLine2
- float totalLengthOfLine2
- float averageLengthOfLine3
- float totalLengthOfLine3
- bool tellerAvailable
- bool teller1
- bool teller2
- bool teller3
- bool emptyLine
- bool emptyLineSpec
- int totalldleTime1
- int idleStart1
- int idleEnd1
- int totalldleTime2
- int idleStart2
- int idleEnd2
- int totalIdleTime3
- int idleStart3
- int idleEnd3

### 3.11.1 Constructor & Destructor Documentation

3.11.1.1 SimulationLinked2::SimulationLinked2 ( int a = 999999 )

Constructor for class SimulationLinked2.

Able to construct a SimulationLinked2 object

Precondition

None

Postcondition

None

None

**Parameters** 

a defaults at 99999

**Exceptions** None Note : None 3.11.1.2 SimulationLinked2::~SimulationLinked2 ( ) Destructor for class SimulationLinked2. Able to destruct a SimulationLinked2 object Precondition None Postcondition None None **Parameters** None **Exceptions** None Note : None 3.11.2 Member Function Documentation 3.11.2.1 void SimulationLinked2::getArrivals ( char \* fileName ) Gets input from file. Gets input from file, specifically, a file of integers in numerical order that will be used for the arrivalQueue Precondition arrivals is empty Postcondition arrivals contains all of the integers from the file Algorithm

For the amount of customers in this simulation, loop goes through the specified file and sets the parameters of

the array of events called arrivals

Parameters	
fileName	
Exceptions	
None	
Note	
: None	
3.11.2.2 int SimulationLinked2::getCurrentLine ( int time )	
Gets the departuring line.	
Gets the line number for the correlating departing line	
Precondition	
None	
None	
Postcondition	
None	
Algorithm	
Compares time to the front event's start time for each line and returns the correlating line number	
Parameters	
time	_
Frankling	
Exceptions None	_
None	_
Note	
: None	
. None	
3.11.2.3 int SimulationLinked2::getShortestLine ( )	
Gets the shortest line.	
Gets the shortest line out of three lines	
Precondition	
shortestLine defaults at 1	
Postcondition	
if any line length is less than 1, shortestLine gets that line number	
Algorithm	
see precondition and postcondition also sets emptyLine to true if either of the three lines are empty	

Parameters
None
Exceptions
None
Note
: None
3.11.2.4 void SimulationLinked2::getTellerAvailability ( int tellerNumber, int time )
Sets each teller's availability.
Sets each teller's availability and sets the general tellerAvailable boolean based on the three tellers' individual values. Also starts and ends idle time for each.
Precondition
teller1, 2, and 3, as well as tellerAvailability
Postcondition
preconditions get set based on parameters
Algorithm
Depending on the tellerNumber, sets correlating teller's data based on time;
Parameters
teller-
Number,time
Exceptions
None
Note
: None
3.11.2.5 void SimulationLinked2::processArrival ( LinkedListQueue < Event > & arrivalQueue, LinkedListQueue < Event > & bankLine, LinkedListQueue < int > & departureQueue )

processes an arrival

dequeus the arrivalQueue, enqueues the eventQueue, enqueues the bankLine if necessary otherwise it enqueues the departureQueue

### Precondition

tellerAvailable will either be available or unavailable

### Postcondition

tellerAvailable will be unavailable if it was available

# Algorithm

First, it dequeues from the arrivalQueue and then enqueues that on to the eventQueue. Then it creates a customer and sets its arrival time, which is necessary for tracking its wait time. If the line is empty and a teller is available, then wait time is 0, determines the departure time, enqueues that on to departureQueue, ends the idle time for the teller, and sets tellerAvailable to false. Otherwise it enqueues the customer's arrival onto the correct bankLine.

#### **Parameters**

	see	details for parameter specifications
Exceptions		
		None

#### Note

: None

3.11.2.6 void SimulationLinked2::processDeparture ( LinkedListQueue < Event > & arrivalQueue, LinkedListQueue < Event > & eventQueue, LinkedListQueue < Event > & bankLine, LinkedListQueue < int > & departureQueue )

processes a departure

dequeus the departure, enqueues the eventQueue, dequeues the bankLine if necessary

# Precondition

tellerAvailable will either be unavailable

#### Postcondition

tellerAvailable will be available if the bankLine is empty

### **Algorithm**

First, it dequeues from the departureQueue and then enqueus that on to the eventQueue. It checks if the correct bankLine is empty. If it is not, then it dequeus the customer from that bank line and determines the customer's wait time. It is then able to determine the departure time of that customer. If the correct bankLine is not empty anymore then the teller's idleTime starts and its availability is set to true.

### **Parameters**

	see	details for parameter specifications	
Eveentions			
Exceptions			
		None	

### Note

: None

3.11.2.7 void SimulationLinked2::simulate ( )

Simulates a bank.

Simulates a bank with three tellers and three lines

#### Precondition

eventQueue, cutsomerQueue, bankLine (3), and departureQueue are empty arrivalQueue is immediately filled with arrivals

#### Postcondition

As arrivalQueue is dequeued, the other queues get enqueued with data and then dequeued according to the arrivals' specification

### Algorithm

My implementation is a little bit different than the book's. Instead of a priority queue, I used three different queues, and arrivalQueue, departureQueue, and an eventQueue. The eventQueue does not do anything except store all the events that occur throughout the algorithm for output at the end of the algorithm. arrivalQueue is instantly enqueued with all of its data and it no longer gets enqueued throughout the algorithm. After enqueing all of the arrivals, it checks what is at the front of the arrivalQueue and at the front of departureQueue. It then uses this data to determine which one it will process. If the arrivalQueue's front time is earlier than the departureQueue's, then it will process an arrival first and vice versa. It also checks if the departureQueue is empty, which it is at the very beginning. It then processes an arrival if this is so, which is always true at the beginning. The multiple bankLine queues, are used to store what customers are still waiting in line by simply enqueing the arrival event when it occurs and then dequeing it once the departure time of the event has been determined. A fifth queue, customerQueue, is only used to store the wait time of each arriving customer. After the first loop finishes, there are still departures that haven't been processed, so it empties the bank line and processes the departures.

### **Parameters**

None		
Exceptions		
	None	

### Note

: Departure upon output will be out of order simply because I did not have enough time to implement more departureQueues to correlate with the multiple tellers. This means that the departureQueue is just out of order because its holding what is technically three different queues. The algorithm is run correctly though and the data is mostly correct as well.

The documentation for this class was generated from the following files:

- · SimulationLinked2.h
- SimulationLinked2.cpp

### 3.12 SimulationLinked3 Class Reference

### **Public Member Functions**

- SimulationLinked3 (int=99999)
  - Constructor for class SimulationLinked3.
- ∼SimulationLinked3 ()

Destructor for class SimulationLinked3.

void getArrivals (char \*)

Gets input from file.

• int getTellerAvailability ()

Determines teller's availability.

• int getTeller ()

Returns which teller is unavailable.

• void simulate ()

Simulates a bank.

void processArrival (LinkedListQueue < Event > &, LinkedListQueue < Event > &, LinkedListQueue < Event > &, LinkedListQueue < int > &)

processes an arrival

void processDeparture (LinkedListQueue< Event > &, LinkedListQueue< Event > &, LinkedListQueue<</li>
 Event > &, LinkedListQueue< int > &)

processes a departure

#### **Private Attributes**

- · bool firstSim
- LinkedListQueue < Customer > customerQueue
- Event customer
- Event \* arrivals
- · int numOfCustomers
- · int numOfEvents
- int currentTime
- int transactionTime
- int maxWaitTime
- · int totalWaitTime
- int maxLengthOfLine
- float averageWaitTime
- float averageLengthOfLine
- · float totalLengthOfLine
- · bool tellerAvailable
- · bool teller1
- bool teller2
- bool teller3
- int totalIdleTime1
- int idleStart1
- int idleEnd1
- int totalIdleTime2
- int idleStart2
- int idleEnd2
- int totalIdleTime3
- int idleStart3
- int idleEnd3

### 3.12.1 Constructor & Destructor Documentation

3.12.1.1 SimulationLinked3::SimulationLinked3 (int a = 99999)

Constructor for class SimulationLinked3.

Able to construct a SimulationLinked3 object

Precondition
None
Desta a undition
Postcondition
None
None
Parameters
a defaults at 99999
a delaults at 99999
Exceptions
None
Note
: None
3.12.1.2 SimulationLinked3::~SimulationLinked3 ( )
Destructor for class SimulationLinked3.
Able to destruct a SimulationLinked3 object
Precondition
None
Postcondition
None
None
Parameters
None
Exceptions
None
Note
: None
3.12.2 Member Function Documentation
3.12.2.1 void SimulationLinked3::getArrivals ( char * fileName )
Gets input from file.
Gets input from file, specifically, a file of integers in numerical order that will be used for the arrivalQueue

48 **CONTENTS** Precondition arrivals is empty Postcondition arrivals contains all of the integers from the file Algorithm For the amount of customers in this simulation, loop goes through the specified file and sets the parameters of the array of events called arrivals **Parameters** fileName **Exceptions** None Note : None 3.12.2.2 int SimulationLinked3::getTeller ( ) Returns which teller is unavailable. None Precondition None Postcondition None Algorithm If teller is unavailable, return teller number **Parameters** None **Exceptions** None Note : None

3.12.2.3 int SimulationLinked3::getTellerAvailability ( )

Determines teller's availability.

Returns which teller is available

Precondition
None
Postcondition
None
Algorithm
Depending on the data of the individual tellers, returns specific teller's number if it is available
Parameters
None
Exceptions
None
Note
: None
3.12.2.4 void SimulationLinked3::processArrival ( LinkedListQueue < Event > & arrivalQueue, LinkedListQueue < Event > & bankLine, LinkedListQueue < int > & departureQueue )
processes an arrival
dequeus the arrivalQueue, enqueues the eventQueue, enqueues the bankLine if necessary otherwise it enqueues the departureQueue
Precondition
tellerAvailable will either be available or unavailable
Postcondition
tellerAvailable will be unavailable if it was available
Algorithm
First, it dequeues from the arrivalQueue and then enqueues that on to the eventQueue. Then it creates a customer and sets its arrival time, which is necessary for tracking its wait time. If the line is empty and a telle is available, then wait time is 0, determines the departure time, enqueues that on to departureQueue, ends the idle time for the teller, and sets tellerAvailable to false. Otherwise it enqueues the customer's arrival onto the correct bankLine.
Parameters
see details for parameter specifications
Exceptions
None
Note
: None

3.12.2.5 void SimulationLinked3::processDeparture ( LinkedListQueue < Event > & arrivalQueue, LinkedListQueue < Event > & eventQueue, LinkedListQueue < Event > & bankLine, LinkedListQueue < int > & departureQueue )

processes a departure

dequeus the departure, enqueues the eventQueue, dequeues the bankLine if necessary

Precondition

tellerAvailable will either be unavailable

#### Postcondition

tellerAvailable will be available if the bankLine is empty

#### **Algorithm**

First, it dequeues from the departureQueue and then enqueus that on to the eventQueue. It checks if the correct bankLine is empty. If it is not, then it dequeus the customer from that bank line and determines the customer's wait time. It is then able to determine the departure time of that customer. If the correct bankLine is not empty anymore then the teller's idleTime starts and its availability is set to true.

#### **Parameters**

see details for parameter specifications
--

#### Exceptions

None	

Note

: None

3.12.2.6 void SimulationLinked3::simulate ( )

Simulates a bank.

Simulates a bank with three tellers and one lines

Precondition

eventQueue, cutsomerQueue, bankLine, and departureQueue are empty arrivalQueue is immediately filled with arrivals

#### Postcondition

As arrivalQueue is dequeued, the other queues get enqueued with data and then dequeued according to the arrivals' specification

### Algorithm

My implementation is a little bit different than the book's. Instead of a priority queue, I used three different queues, and arrivalQueue, departureQueue, and an eventQueue. The eventQueue does not do anything except store all the events that occur throughout the algorithm for output at the end of the algorithm. arrivalQueue is instantly enqueued with all of its data and it no longer gets enqueued throughout the algorithm. After enqueing all of the arrivals, it checks what is at the front of the arrivalQueue and at the front of departureQueue. It then uses this data to determine which one it will process. If the arrivalQueue's front time is earlier than the departureQueue's, then it will process an arrival first and vice versa. It also checks if the departureQueue is empty, which it is at the very beginning. It then processes an arrival if this is so, which is always true at the

4 File Documentation 51

beginning. The bankLine queues is used to store what customers are still waiting in line by simply enqueing the arrival event when it occurs and then dequeing it once the departure time of the event has been determined. A fifth queue, customerQueue, is only used to store the wait time of each arriving customer. After the first loop finishes, there are still departures that haven't been processed, so it empties the bank line and processes the departures.

#### **Parameters**

None	е	
Eventions		
Exceptions		
	None	

### Note

: Departure upon output will be out of order simply because I did not have enough time to implement more departureQueues to correlate with the multiple tellers. This means that the departureQueue is just out of order because its holding what is technically three different queues. The algorithm is run correctly though and the data is mostly correct as well.

The documentation for this class was generated from the following files:

- · SimulationLinked3.h
- SimulationLinked3.cpp

# 4 File Documentation

# 4.1 LinkedListQueue.cpp File Reference

Implementation file for LinkedListQueue class.

```
#include <iostream>
#include <fstream>
#include <string>
#include "Event.h"
#include "Customer.h"
```

### Classes

- struct Node < ItemType >
- class LinkedListQueue < ItemType >

# 4.1.1 Detailed Description

Implementation file for LinkedListQueue class.

Author

Alex Kastanek

Implements all member methods of the LinkedListQueue class

#### Version

1.00 C.S. Student (15 November 2016) Initial development and testing of LinkedListQueue class

Note

This class is derived from one I made last semester (Spring 2016). Any formatting inconsistencies are because of this.

# 4.2 Queue.cpp File Reference

Implementation file for Queue class.

```
#include <iostream>
#include <fstream>
#include <string>
#include "Event.h"
#include "Customer.h"
```

### Classes

class Queue < ItemType >

# 4.2.1 Detailed Description

Implementation file for Queue class.

**Author** 

Alex Kastanek

Implements all member methods of the Queue class

Version

1.00 C.S. Student (15 November 2016) Initial development and testing of Queue class

Note

This class is derived from one I made last semester (Spring 2016). Any formatting inconsistencies are because of this.

### 4.3 radixSort.h File Reference

Definition file for radixSort class.

```
#include <iostream>
```

### Classes

class RadixSort

### 4.3.1 Detailed Description

Definition file for radixSort class.

**Author** 

Alex Kastanek

Specifies all member methods of the radixSort class

Version

1.00 C.S. Student (1 November 2016) Initial development and testing of radixSort class

Note

None

# 4.4 Simulation1.cpp File Reference

Implementation file for Simulation1 class.

```
#include "Simulation1.h"
```

### 4.4.1 Detailed Description

Implementation file for Simulation1 class.

**Author** 

Alex Kastanek

Implements all member methods of the Simulation1 class

Version

1.00 C.S. Student (15 November 2016) Initial development and testing of Simulation1 class

Note

Requires Simulation1.h None

# 4.5 Simulation1.h File Reference

Definition file for Simulation1 class.

```
#include <iostream>
#include <fstream>
#include <string>
#include <stdlib.h>
#include <time.h>
#include "Event.h"
#include "Queue.cpp"
#include "Customer.h"
```

#### Classes

class Simulation1

#### 4.5.1 Detailed Description

Definition file for Simulation1 class.

**Author** 

Alex Kastanek

Specifies all member methods of the Simulation1 class

Version

1.00 C.S. Student (15 November 2016) Initial development and testing of Simulation1 class

Note

None

# 4.6 Simulation2.cpp File Reference

Implementation file for Simulation2 class.

```
#include "Simulation2.h"
```

# 4.6.1 Detailed Description

Implementation file for Simulation2 class.

**Author** 

Alex Kastanek

Implements all member methods of the Simulation2 class

Version

1.00 C.S. Student (15 November 2016) Initial development and testing of Simulation2 class

Note

Requires Simulation2.h None

# 4.7 Simulation2.h File Reference

Definition file for Simulation2 class.

```
#include <iostream>
#include <fstream>
#include <string>
#include <stdlib.h>
#include <time.h>
#include "Event.h"
#include "Queue.cpp"
#include "Customer.h"
```

Classes

class Simulation2

#### 4.7.1 Detailed Description

Definition file for Simulation2 class.

Author

Alex Kastanek

Specifies all member methods of the Simulation2 class

Version

1.00 C.S. Student (15 November 2016) Initial development and testing of Simulation2 class

Note

None

# 4.8 Simulation3.cpp File Reference

Implementation file for Simulation3 class.

```
#include "Simulation3.h"
```

### 4.8.1 Detailed Description

Implementation file for Simulation3 class.

**Author** 

Alex Kastanek

Implements all member methods of the Simulation3 class

Version

1.00 C.S. Student (15 November 2016) Initial development and testing of Simulation3 class

Note

Requires Simulation3.h None

# 4.9 Simulation3.h File Reference

Definition file for Simulation3 class.

```
#include <iostream>
#include <fstream>
#include <string>
#include <stdlib.h>
#include <time.h>
#include "Event.h"
#include "Queue.cpp"
#include "Customer.h"
```

#### Classes

class Simulation3

#### 4.9.1 Detailed Description

Definition file for Simulation3 class.

Author

Alex Kastanek

Specifies all member methods of the Simulation3 class

Version

1.00 C.S. Student (15 November 2016) Initial development and testing of Simulation3 class

Note

None

# 4.10 SimulationLinked1.cpp File Reference

Implementation file for SimulationLinked1 class.

```
#include "SimulationLinked1.h"
```

# 4.10.1 Detailed Description

Implementation file for SimulationLinked1 class.

**Author** 

Alex Kastanek

Implements all member methods of the SimulationLinked1 class

Version

1.00 C.S. Student (15 November 2016) Initial development and testing of SimulationLinked1 class

Note

```
Requires SimulationLinked1.h None
```

# 4.11 SimulationLinked1.h File Reference

Definition file for SimulationLinked1 class.

```
#include <iostream>
#include <fstream>
#include <string>
#include <stdlib.h>
#include <time.h>
#include "Event.h"
#include "LinkedListQueue.cpp"
#include "Customer.h"
```

Classes

· class SimulationLinked1

#### 4.11.1 Detailed Description

Definition file for SimulationLinked1 class.

**Author** 

Alex Kastanek

Specifies all member methods of the SimulationLinked1 class

Version

1.00 C.S. Student (15 November 2016) Initial development and testing of SimulationLinked1 class

Note

None

# 4.12 SimulationLinked2.cpp File Reference

Implementation file for SimulationLinked2 class.

```
#include "SimulationLinked2.h"
```

# 4.12.1 Detailed Description

Implementation file for SimulationLinked2 class.

**Author** 

Alex Kastanek

Implements all member methods of the SimulationLinked2 class

Version

1.00 C.S. Student (15 November 2016) Initial development and testing of SimulationLinked2 class

Note

Requires SimulationLinked2.h None

# 4.13 SimulationLinked2.h File Reference

Definition file for SimulationLinked2 class.

```
#include <iostream>
#include <fstream>
#include <string>
#include <stdlib.h>
#include <time.h>
#include "Event.h"
#include "LinkedListQueue.cpp"
#include "Customer.h"
```

#### Classes

• class SimulationLinked2

#### 4.13.1 Detailed Description

Definition file for SimulationLinked2 class.

**Author** 

Alex Kastanek

Specifies all member methods of the SimulationLinked2 class

Version

1.00 C.S. Student (15 November 2016) Initial development and testing of SimulationLinked2 class

Note

None

# 4.14 SimulationLinked3.cpp File Reference

Implementation file for SimulationLinked3 class.

```
#include "SimulationLinked3.h"
```

### 4.14.1 Detailed Description

Implementation file for SimulationLinked3 class.

**Author** 

Alex Kastanek

Implements all member methods of the SimulationLinked class

Version

1.00 C.S. Student (15 November 2016) Initial development and testing of SimulationLinked3 class

Note

Requires SimulationLinked3.h None

# 4.15 SimulationLinked3.h File Reference

Definition file for SimulationLinked3 class.

```
#include <iostream>
#include <fstream>
#include <string>
#include <stdlib.h>
#include <time.h>
#include "Event.h"
#include "LinkedListQueue.cpp"
#include "Customer.h"
```

Classes

• class SimulationLinked3

# 4.15.1 Detailed Description

Definition file for SimulationLinked3 class.

Author

Alex Kastanek

Specifies all member methods of the SimulationLinked3 class

Version

1.00 C.S. Student (15 November 2016) Initial development and testing of SimulationLinked3 class

Note

None

# Index

$\sim$ LinkedListQueue	isEmpty
LinkedListQueue, 5	LinkedListQueue, 7
$\sim$ Queue	Queue, 13
Queue, 11	isFull
~RadixSort	LinkedListQueue, 8
RadixSort, 17	Queue, 14
~Simulation1	Quodo, TT
Simulation1, 19	LinkedListQueue
~Simulation?	~LinkedListQueue, 5
Simulation2, 24	clear, 6
~Simulation3	dequeue, 6
Simulation3, 30	enqueue, 6
$\sim$ SimulationLinked1	getLength, 7
SimulationLinked1, 36	isEmpty, 7
$\sim$ SimulationLinked2	isFull, 8
SimulationLinked2, 41	LinkedListQueue, 5
$\sim$ SimulationLinked3	LinkedListQueue, 5
SimulationLinked3, 47	peekFront, 8
,	print, 9
clear	printToFile, 9
LinkedListQueue, 6	LinkedListQueue < ItemType >, 4
Customer, 3	LinkedListQueue.cpp, 51
33333.,	
dequeue	Node < ItemType >, 10
LinkedListQueue, 6	riodo ( nomigeo > , ro
Queue, 12	peekFront
Queue, 12	LinkedListQueue, 8
enqueue	Queue, 14
LinkedListQueue, 6	print
	•
Queue, 12	LinkedListQueue, 9
Event, 3	Queue, 15
getArrivals	printToFile
Simulation1, 20	LinkedListQueue, 9
•	Queue, 15
Simulation2, 25	processArrival
Simulation3, 31	Simulation1, 20
SimulationLinked1, 36	Simulation2, 27
SimulationLinked2, 41	Simulation3, 32
SimulationLinked3, 47	SimulationLinked1, 37
getCurrentLine	SimulationLinked2, 43
Simulation2, 25	SimulationLinked3, 49
SimulationLinked2, 42	processDeparture
getLength	Simulation1, 21
LinkedListQueue, 7	Simulation2, 27
Queue, 13	Simulation3, 33
getShortestLine	SimulationLinked1, 37
Simulation2, 26	SimulationLinked2, 44
SimulationLinked2, 42	SimulationLinked3, 49
getTeller	OfficiationEfficace, 40
Simulation3, 31	Queue
SimulationLinked3, 48	~Queue, 11
getTellerAvailability	dequeue, 12
- ·	•
Simulation2, 26	enqueue, 12
Simulation3, 32	getLength, 13
SimulationLinked2, 43	isEmpty, 13
SimulationLinked3, 48	isFull, 14

INDEX 61

peekFront, 14 print, 15	SimulationLinked1, 35 SimulationLinked1, 35
printToFile, 15	SimulationLinked1.cpp, 56
Queue, 11	SimulationLinked1.h, 56
Queue< ItemType >, 10	SimulationLinked2, 39
Queue.cpp, 52	~SimulationLinked2, 4
Quous.opp, 02	getArrivals, 41
RadixSort, 16	getCurrentLine, 42
$\sim$ RadixSort, 17	getShortestLine, 42
RadixSort, 17	getTellerAvailability, 43
RadixSort, 17	processArrival, 43
sort, 18	processDeparture, 44
radixSort.h, 52	simulate, 44
	SimulationLinked2, 40
simulate	SimulationLinked2, 40
Simulation1, 22	SimulationLinked2.cpp, 57
Simulation2, 28	SimulationLinked2.h, 57
Simulation3, 33	SimulationLinked3, 45
SimulationLinked1, 38	$\sim$ SimulationLinked3, 47
SimulationLinked2, 44	getArrivals, 47
SimulationLinked3, 50	getTeller, 48
Simulation1, 18	getTellerAvailability, 48
~Simulation1, 19	processArrival, 49
getArrivals, 20 processArrival, 20	processDeparture, 49
processDeparture, 21	simulate, 50
simulate, 22	SimulationLinked3, 46
Simulation1, 19	SimulationLinked3, 46
Simulation1.cpp, 53	SimulationLinked3.cpp, 58
Simulation1.h, 53	SimulationLinked3.h, 58
Simulation2, 23	sort
~Simulation2, 24	RadixSort, 18
getArrivals, 25	
getCurrentLine, 25	
getShortestLine, 26	
getTellerAvailability, 26	
processArrival, 27	
processDeparture, 27	
simulate, 28	
Simulation2, 24	
Simulation2.cpp, 54	
Simulation2.h, 54	
Simulation3, 29	
$\sim$ Simulation3, 30	
getArrivals, 31	
getTeller, 31	
getTellerAvailability, 32	
processArrival, 32	
processDeparture, 33	
simulate, 33	
Simulation3, 30	
Simulation3.cpp, 55	
Simulation3.h, 55	
SimulationLinked1, 34	
~SimulationLinked1, 36 getArrivals, 36	
processArrival, 37	
processDeparture, 37	
simulate, 38	
, <del>-</del> -	