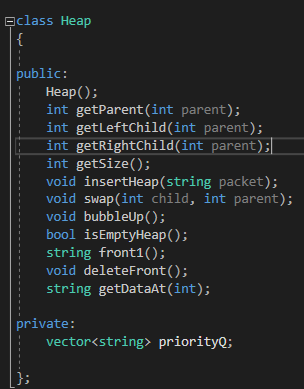
Rushi Patel 100615230 Yash Patel 100621177 Jeet Desai 100635399

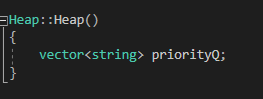
Algorithms and Data structure final project Report

1. Implement the priority Q’s using a heaps

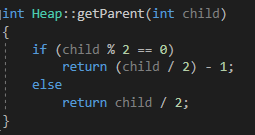
Header class to define functions used to create a heap



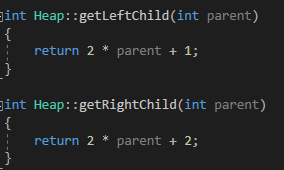
Constructor which defines a vector which was used to hold all the packets from FIFO queue and sort them into priority queues 100, and 101 according to their priority



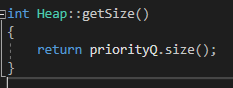
Parent function to get parent node from the tree



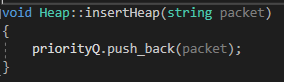
These functions are used to get left child and right child of a parent node



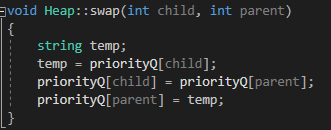
getSize function returns priority queue size



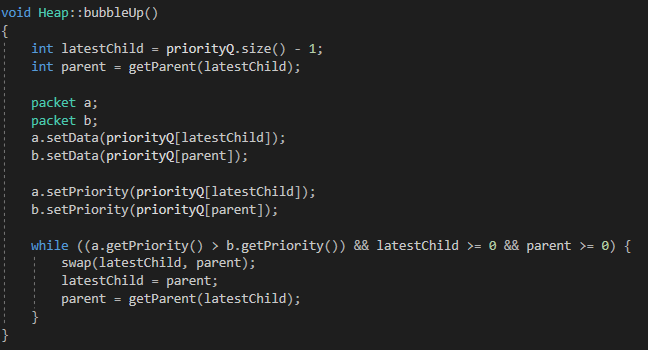
Insert function takes a packet as a parameter and pushes it back in the priority queue



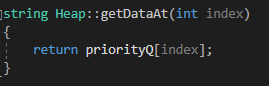
Swap function takes two parameters child and parent to swap parent with left or right child if any of them is greater than parent node.



Bubbleup function takes latest child and keeps comparing with the parent node until latest child reach to the point where it is equal to the highest value or it is first in the queue. Packet with priority 2 will keep swapping until it is compared with a node which is holding a packet with priority 2.

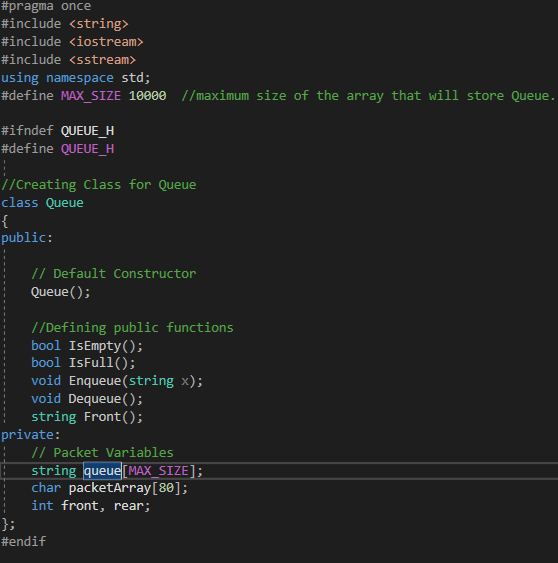


This function takes in an index as a parameter and returns a value from that specific index

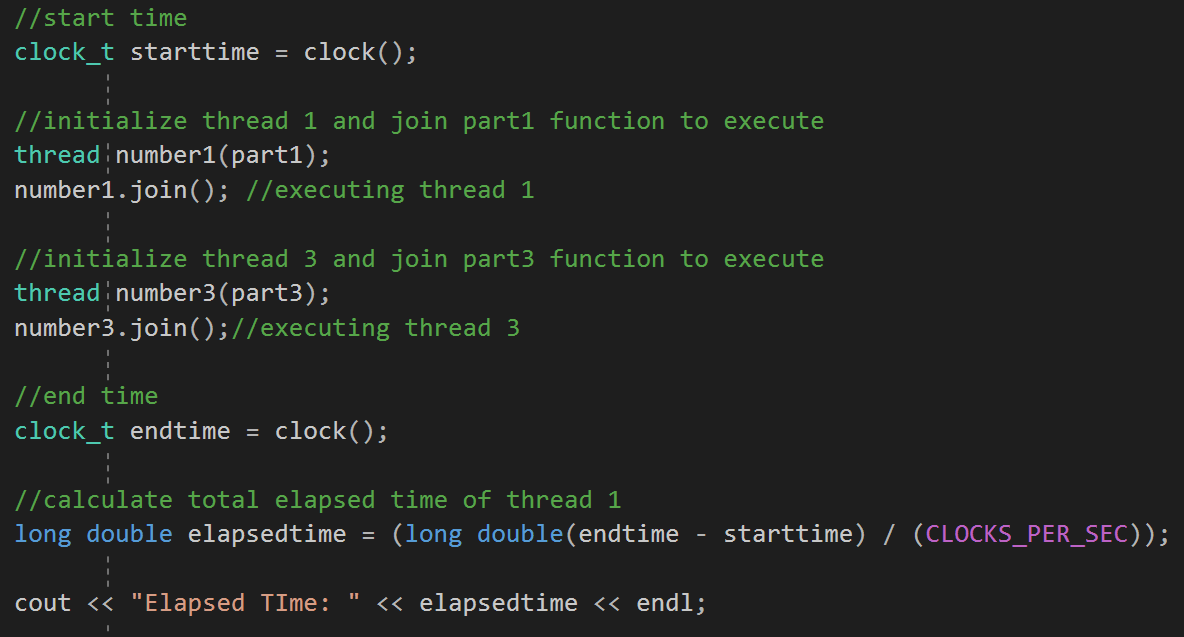


**Queue initializations**

Array-based Implementation was used to create packet class to handle packets going in/out of queue



2. Evaluate the performance for different sizes of the input files.



|  |  |
| --- | --- |
| Packet Size | Run Time (seconds) |
| 10 | 10.526 |
| 20 | 19.86 |
| 30 | 29.166 |
| 40 | 37.606 |
| 50 | 44.203 |

The reason why we took such packet sizes and limited till only 50 packets is because of the 1 second delay in thread 3 when everytime a packet from a queue is read. If we were to test with larger number of packets there would obviously be a larger run-time. So with this analysis we can assume that, the run-time would be approximate of how many packets are in the file.

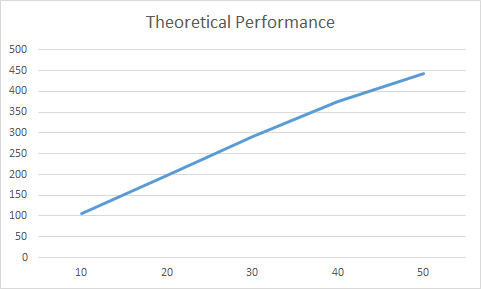
Run-Time performance graph



3. Compare the run-time performance to the theoretical performance

|  |  |  |
| --- | --- | --- |
| Packet Size | Run-Time Performance | Theoretical Performance  T(n) = cop.C(n) |
| 10 | 10.526 | T(n) = 10 \* 10.526 = 105.26 |
| 20 | 19.86 | T(n) = 10 \* 19.86 = 198.6 |
| 30 | 29.166 | T(n) = 10 \* 29.166 = 291.66 |
| 40 | 37.606 | T(n) = 10 \* 37.606 = 376.06 |
| 50 | 44.203 | T(n) = 10 \* 44.203 = 442.03 |

Theoretical performance graph



Run-Time and Theoretical Time comparison

