Austin W. & Chris R

LA7 Psudocode

Answer to B: public class PriorityQueue<E extends Comparable<E>>

Answer to C: public class Request implements Comparable<Request>

The parameter is a the compareTo method represents a request object to compare with

And returns 1 for higher priority, -1 for lower priority and 0 for same priority

Class Node {

Declare field for next Node

Declare field for generic data

Node(data) {

Set data to equal parameter

Set next node to null

Node(data, Node next)

Set data to equal data parameter

Set next node to equal next node paramter

Getter and Setters for data value and Node next

Class LinkedList{

isEmpty(){

if head node is null return true

else return false

add(item){

if list is empty

create new node with item data

set new node and tail to head

else

set next node from tail to new node

new tail is new node

get(int position)

create a copy of head node

iterate through until at position parameter

return data at position

size() {

create copy of head node

iterate through until next node is null

return how many times the loop, looped

class Request {

declare fields needed

Request(data parameters) {

Save each value to a field grabbed from input file

compareTo(Request req) {

if(this value > request object value)

return 1 //meaning higher priority

else if (this value < request object value

return -1 //meaning lower priority

else

repeat same for next priority value

else

repeat same for next priority value

else

return 0 //meaning same priority

yearsFromGraduation(string level){

switch(level)

check each case “freshman, sophomore, etc”

return 0-3 for years left to graduation

if none match return -1

GPA\_Cal(double[][] GPA\_Array){

Iterate through array

Sum = gpa\*credit

Return sum/sum of credit

Field getters and setters{

Class PriortyQueue<E extends Comparable<E>>{

isEmpty(){

if front == null return true

else return false

enqueue(E data){

create new node with data

copy head node

if queue is empty

front = new node

rear = front

if only has one node

if has priority over existing node

existingnode.next = newnode

rear = existing node

front = newnode

else

exsistingnode.next = newnode

rear = newnode

if queue has multiple nodes

if new node has higher priority than front

newnode.next = rear

front = newnode

if same priority

newnode.next = exisitingnode.next

exisitngnode.next = newnode

if lower priority than front

loop while not sorted and nextnode is not rear

if higher priority than existing node

newnode.next = existingnode.next

existingnode.next = new node

if same priority as existing node

newnode.next = exisisitng.next.next

existing.next = new node

sorted = true

if lower priority as exisisintg

move to next node

dequeue(){

make copies of front and rear nodes

front = front.next

return front.data

Qprint(){

Copy front node

If front = null

Print queue is empty

Else

While copy node is not rear

Print node data content

Copy = copy.next