Austin Wojciechowski & Chris Reyes

**LA7 Pseudocode**

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

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

The parameter T in the compareTo method represents the object to compare to the class object

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**{

Variables

Head node

Tail node

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

student name, dept, level

course dept, number

GPA array

GPA final

Years to grad

Student department same as course

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

toString()

return request@ + obj id

Field getters and setters{

Class **PriorityQueue**<E extends Comparable<E>>{

Variables

Front node

Rear node

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 = frontcopy

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 next

newnode.next = existingnode.next

existingnode.next = new node

sorted = true

if same priority as existing node next

newnode.next = exisisitng.next.next

existing.next = new node

sorted = true

if lower priority than existing node next

move to next node

if still not sorted and nextnode is rear

if higher priority than existing node

newnode.next = existingnode.next

existingnode.next = new node

else

existingnode.next = new node

rear = new node

dequeue(){

make copies of front and rear nodes

front = front.next

if front was only node

rear = front

return front.data

Qprint(){

Copy front node

If front = null

Print queue is empty

Else

While copy node is not rear

Print node data toString

Copy = copy.next

Print rear node toString

Class **Controller** implements IController

* Variables
  + Queue of requests
  + Course list
  + Course file reader
  + Request file reader
* Controller([variables])
  + sets each variable to corresponding input parameter
* readCourseFile()
  + try
    - while current line being read is not null
      * print current line
      * split line into string array of line components(course dept, course number, capacity)
      * create a new course using line components as parameters
      * add course to linked list of courses
    - close the file reader
  + catch(IOException)
    - printStackTrace
* readRequestFile()
  + try
    - while current line being read is not null
      * print current line
      * split line into string array of line components
        + (student name, student dept, student level, course dept, course number, GPAs, credits)
        + GPA\_array[][]

Row 0 = GPA in one class

Row 1 = Credit in same class

* + - * create a new request using line components and GPA\_array as parameters
      * add request to priority queue of requests
    - close the file reader
  + catch(IOException)
    - printStackTrace
* addRequest(Request req)
  + enqueues req to the priority queue
* processRequests()
  + prints out queue before processing requests
  + processes each request in a while loop
    - while queue is not empty
      * dequeue the top priority request and save data as a var
      * find requested course using getCourse()
      * print the request id
      * if the course is not full
        + reduce course capacity by 1
        + add student to course class list
        + print that the student has been registered
      * else
        + print that the student cannot be registered
  + reprint the now empty queue to show it’s empty
* getCourse(String courseDept, int courseNumber)
  + searches through linked list of courses for the course with the same dept and course number
  + if it is found, returns the course
  + otherwise, returns null
* printClassList
  + for each course in the list of courses
    - prints class dept and number
    - prints the class list of students

Class **Course** implements ICourse

* variables
  + course dept
  + course num
  + capacity
  + list of students
* Course([variables])
  + sets each variable to corresponding input parameter
* isFull()
  + checks if capacity is 0
    - if so, full
    - else, not full
* addStudent(String name)
  + add input name to list of students in the class list
* printClassList()
  + check if class is empty
    - if not, print each student in the class list
    - else, print that the class is empty
* getters and setters for each variable