**CS300 MOD 3 Milestone Pseudocode and Runtime Complexities**

Pseudocode

**BEGIN**

STRUCT Course

INIT String courseNumber

INIT String courseName

INIT vector <string type> preReq

END STRUCT

FUNCTION courseSearch

PASS IN: vector <Course type>& courses, String courseNum

INIT integer coursesSize as size of courses

INIT integer index as 0

WHILE index is less than courseSize THEN

IF courses[index].courseNumber equals courseNum THEN

RETURN index

END IF

INCREMENT index

END WHILE

RETURN -1

END FUNCTION

FUNCTION printCourse

PASS IN: vector <Course type>& courses, String courseNum

INIT integer courseStatus as CALL courseSearch with courses, courseNum

IF courseStatus equals -1 THEN

OUTPUT “Course Information was not found within schedule.”

RETURN

ELSE THEN

OUTPUT “Course Information found on the following row in the schedule: “

OUTPUT courseStatus + 1

OUTPUT NEWLINE

OUTPUT courses[courseStatus].courseNumber

OUTPUT “, “

OUTPUT courses[courseStatus].courseName

INIT integer length as size of courses[courseStatus].preReq

INIT integer index as 0

FOR index is less than length THEN

OUTPUT “, “

OUTPUT courses[courseStatus].preReq[index]

IF index does NOT equal (length minus 1) THEN

OUTPUT ","

END IF

INCREMENT index

END FOR

END IF

END FUNCTION

FUNCTION loadVector

PASS IN: String fileName

INIT vector <Course type> courses

INIT String line

INIT String parser

INIT vector<String type> tempVector

OUTPUT "Opening the following file: "

OUTPUT fileName

OUTPUT NEWLINE

OPEN fileName

IF fileName does NOT open THEN

OUTPUT "File cannot be opened"

CLOSE fileName

ELSE THEN

WHILE NOT end of file THEN

EMPTY tempVector

GET LINE with fileName, line

IF length of line equals 0 THEN

OUTPUT “No data in line”

END IF

SWITCH (count of "," in line)

CASE 0 THEN

OUTPUT "Invalid line"

BREAK

CASE 1 THEN

WHILE getline(line, parser, “,”) THEN

PUSH BACK parser into tempVector

END WHILE

ASSIGN course.courseNumber with tempVector[0]

ASSIGN course.courseName with tempVector[1]

PUSH BACK course into courses

BREAK

DEFAULT THEN

WHILE getline(line, parser, “,”) THEN

PUSH BACK parser into tempVector

END WHILE

ASSIGN course.courseNumber with tempVector[0]

ASSIGN course.courseName with tempVector[1]

INIT integer preReqCount with 2

WHILE preReqCount is less than size of tempVector THEN

INIT integer courseNumCount with count of tempVector[preReqCount] in fileName

IF courseNumCount > 1 THEN

ASSIGN course.preReq with tempVector[preReqCount]

END IF

INCREMENT preReqCount

END WHILE

PUSH BACK course into courses

BREAK

END CASE

END WHILE

CLOSE fileName

END IF

END FUNCTION

FUNCTION partition

PASS IN: vector <Course type>& courses, integer type begin, integer type end

INIT integer type low with begin

INIT integer type high with end

INIT boolean type done as false

INIT integer type mid as low plus (high minus low) divided by two

INIT string type pivot as courses[mid].courseNumber

INIT string type temp as -1

WHILE NOT done THEN

WHILE courses[low].courseNumber is less than pivot THEN

INCREMENT low

END WHILE

WHILE pivot is less than courses[high].courseNumber THEN

DECREMENT high

END WHILE

IF low is greater than or equal to high THEN

ASSIGN done with true

ELSE THEN

ASSIGN temp with courses[low]

ASSIGN courses[low] with courses[high]

ASSIGN courses[high] with temp

INCREMENT low

DECREMENT high

END IF

END WHILE

RETURN high

END FUNCTION

FUNCTION quickSort

PASS IN: vector <Course type>& courses, integer type begin, integer type end

INIT integer type mid as 0

IF begin is greater than or equal to end THEN

RETURN

END IF

ASSIGN mid with CALL to partition with courses, begin, end

CALL quickSort with courses, begin, mid

CALL quicksort with courses, mid plus one, end

END FUNCTION

MAIN

INIT integer type choice as 0

INIT string type csvName as the name of the CSV file being loaded

INIT vector <Course Type> courses

WHILE choice is NOT 9 THEN

DISPLAY “Menu:”

DISPLAY NEWLINE

DISPLAY “1. Load Courses”

DISPLAY NEWLINE

DISPLAY “2. Print all Courses”

DISPLAY NEWLINE

DISPLAY “3. Find Course”

DISPLAY NEWLINE

ASSIGN choice with input from the user.

SWITCH (choice)

CASE 1 THEN

ASSIGN courses with CALL to loadVector with csvName

BREAK

CASE 2 THEN

INIT integer type coursesSize as the size of courses

INIT integer type index1 as 0

INIT integer type index2 as 0

WHILE index1 is less than coursesSize THEN

` OUTPUT courses[index1].courseNumber

OUTPUT “, “

OUTPUT courses[index1].courseName

INIT integer length as size of courses[index1].preReq

WHILE index2 is less than length THEN

OUTPUT “, “

OUTPUT courses[index1].preReq[index2]

IF index2 does NOT equal (length minus 1) THEN

OUTPUT ","

END IF

INCREMENT index2

END WHILE

INCREMENT index1

END WHILE

BREAK

CASE 3 THEN

INIT string type courseNum

DISPLAY “Please input the Course Number you are searching for.

DISPLAY NEWLINE

INPUT user inputs a Course Number

ASSIGN courseNum with INPUT

CALL printCourse with courses, courseNum

BREAK

CASE 9 THEN

EXIT

END SWITCH

END WHILE

END MAIN

**END**

Analysis

**courseSearch**

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| INIT coursesSize as size of courses | 1 | 1 | 1 |
| INIT index as 0 | 1 | 1 | 1 |
| WHILE index is less than courseSize THEN | 1 | N | N, 1 |
| IF courses[index].courseNumber equals courseNum THEN | 1 | 1 | 1 |
| RETURN index | 1 | 1 | 1 |
| INCREMENT index | 1 | 1 | 1 |
| END WHILE | 1 | 1 | 1 |
| RETURN -1 | 1 | 1 | 1 |
| **Total Cost** | | | Best Case: 2 + 3 = 5  Worst Case: 2+ N(3) + 2 = 4 +3N |
| **Runtime** | | | Best case: O(1)  Worst case:  O(N) |

**printCourse**

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| INIT integer courseStatus as CALL courseSearch with courses, courseNum | 1 | 1 | Best case: 5  Worst case:  2 + N(3) + 2 = 4+ 3N |
| IF courseStatus equals -1 THEN | 1 | 1 | 1 |
| OUTPUT “Course Information was not found within schedule.” | 1 | 1 | 1 |
| RETURN | 1 | 1 | 1 |
| ELSE THEN | 1 | 1 | 1 |
| OUTPUT “Course Information found on the following row in the schedule: “ | 1 | 1 | 1 |
| OUTPUT courseStatus + 1 | 1 | 1 | 1 |
| OUTPUT NEWLINE | 1 | 1 | 1 |
| OUTPUT courses[courseStatus].courseNumber | 1 | 1 | 1 |
| OUTPUT “, “ | 1 | 1 | 1 |
| OUTPUT courses[courseStatus].courseName | 1 | 1 | 1 |
| OUTPUT “, ” | 1 | 1 | 1 |
| INIT integer length as size of courses[courseStatus].preReq | 1 | 1 | 1 |
| INIT integer index as 0 | 1 | 1 | 1 |
| FOR index in length THEN | 1 | N | N, 1 |
| OUTPUT courses[courseStatus].preReq[index] | 1 | 1 | 1 |
| IF index does NOT equal (length minus 1) THEN | 1 | 1 | 1 |
| OUTPUT "," | 1 | 1 | 1 |
| INCREMENT index | 1 | 1 | 1 |
| END FOR | 1 | 1 | 1 |
| **Total Cost** | | | Best Case: 8  Worst Case: 16 + 3N + 5N + 1 = 17 + 8N |
| **Runtime** | | | Best case: O(1)  Worst case:  O(N) |

**loadVector**

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| INIT vector <Course type> courses | 1 | 1 | 1 |
| INIT String line | 1 | 1 | 1 |
| INIT String parser | 1 | 1 | 1 |
| INIT vector<String type> tempVector | 1 | 1 | 1 |
| OUTPUT "Opening the following file: " | 1 | 1 | 1 |
| OUTPUT fileName | 1 | 1 | 1 |
| OUTPUT NEWLINE | 1 | 1 | 1 |
| OPEN fileName | 1 | 1 | 1 |
| IF fileName does NOT open THEN | 1 | 1 | 1 |
| OUTPUT "File cannot be opened" | 1 | 1 | 1 |
| CLOSE fileName | 1 | 1 | 1 |
| ELSE THEN | 1 | 1 | 1 |
| WHILE NOT end of file THEN | 1 | N | N, 1 |
| EMPTY tempVector | 1 | 1 | 1 |
| GET LINE with fileName, line | 1 | 1 | 1 |
| IF length of line equals 0 THEN | 1 | 1 | 1 |
| OUTPUT “No data in line” | 1 | 1 | 1 |
| SWITCH (count of "," in line) | 1 | 1 | 1 |
| CASE 0 THEN | 1 | 1 | 1 |
| OUTPUT "Invalid line" | 1 | 1 | 1 |
| BREAK | 1 | 1 | 1 |
| CASE 1 THEN | 1 | 1 | 1 |
| WHILE getline(line, parser, “,”) THEN | 1 | N | N, 1 |
| PUSH BACK parser into tempVector | 1 | 1 | 1 |
| END WHILE | 1 | 1 | 1 |
| ASSIGN course.courseNumber with tempVector[0] | 1 | 1 | 1 |
| ASSIGN course.courseName with tempVector[1] | 1 | 1 | 1 |
| PUSH BACK course into courses | 1 | 1 | 1 |
| BREAK | 1 | 1 | 1 |
| DEFAULT THEN | 1 | 1 | 1 |
| WHILE getline(line, parser, “,”) THEN | 1 | N | N, 1 |
| PUSH BACK parser into tempVector | 1 | 1 | 1 |
| END WHILE | 1 | 1 | 1 |
| ASSIGN course.courseNumber with tempVector[0] | 1 | 1 | 1 |
| ASSIGN course.courseNumber with tempVector[1] | 1 | 1 | 1 |
| INIT integer preReqCount with 2 | 1 | 1 | 1 |
| WHILE preReqCount is less than size of tempVector THEN | 1 | N | N, 1 |
| INIT integer courseNumCount with count of tempVector[preReqCount] in fileName | 1 | 1 | 1 |
| IF courseNumCount > 1 THEN | 1 | 1 | 1 |
| ASSIGN course.preReq with tempVector[preReqCount] | 1 | 1 | 1 |
| INCREMENT preReqCount | 1 | 1 | 1 |
| END WHILE | 1 | 1 | 1 |
| PUSH BACK course into courses | 1 | 1 | 1 |
| BREAK | 1 | 1 | 1 |
| END WHILE | 1 | 1 | 1 |
| CLOSE fileName | 1 | 1 | 1 |
| **Total Cost** | | | Best Case: 11  Worst Case: 10 + N(8 +N(2) + 4 + N(5) + 3) + 2 = 12 + 22N^2 |
| **Runtime** | | | Best case: O(1)  Worst case:  O(N^2) |

**partition**

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| INIT integer type low with begin | 1 | 1 | 1 |
| INIT integer type high with end | 1 | 1 | 1 |
| INIT boolean type done as false | 1 | 1 | 1 |
| INIT integer type mid as low plus (high minus low) divided by two | 1 | 1 | 1 |
| INIT string type pivot as courses[mid].courseNumber | 1 | 1 | 1 |
| INIT string type temp as -1 | 1 | 1 | 1 |
| WHILE NOT done THEN | 1 | N | N, 1 |
| WHILE courses[low].courseNumber is less than pivot THEN | 1 | N | N, 1 |
| INCREMENT low | 1 | 1 | 1 |
| END WHILE | 1 | 1 | 1 |
| WHILE pivot is less than courses[high].courseNumber THEN | 1 | N | N, 1 |
| DECREMENT high | 1 | 1 | 1 |
| END WHILE | 1 | 1 | 1 |
| IF low is greater than or equal to high THEN | 1 | 1 | 1 |
| ASSIGN done with true | 1 | 1 | 1 |
| ELSE THEN | 1 | 1 | 1 |
| ASSIGN temp with courses[low] | 1 | 1 | 1 |
| ASSIGN courses[low] with courses[high] | 1 | 1 | 1 |
| ASSIGN courses[high] with temp | 1 | 1 | 1 |
| INCREMENT low | 1 | 1 | 1 |
| DECREMENT high | 1 | 1 | 1 |
| END WHILE | 1 | 1 | 1 |
| RETURN high | 1 | 1 | 1 |
| **Total Cost** | | | Best Case: 6 + N(5) + 2 = 8 + 5N  Worst Case: 6 + N(1 + 2N + 1 + 2N + 1 + 7) + 2 = 8 + 14N^2 |
| **Runtime** | | | Best case: O(N)  Worst case:  O(N^2) |

**quickSort**

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| INIT integer type mid as 0 | 1 | 1 | 1 |
| IF begin is greater than or equal to end THEN | 1 | 1 | 1 |
| RETURN | 1 | 1 | 1 |
| ASSIGN mid with CALL to partition with courses, begin, end | 1 | 1 | Best Case: 6 + N(5) + 2 = 8 + 5N  Worst Case: 6 + N(1 + 2N + 1 + 2N + 1 + 7) + 2 = 8 + 14N^2 |
| CALL quickSort with courses, begin, mid | 1 | 1 | Best Case: log (n) Worst Case: N |
| CALL quicksort with courses, mid plus one, end | 1 | 1 | Best Case: log (n) Worst Case: N |
| **Total Cost** | | | Best Case: 3  Worst Case: 10 + 14N^2 |
| **Runtime** | | | Best case: O(1)  Worst case: O(N^2) |

**MAIN**

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| INIT integer type choice as 0 | 1 | 1 | 1 |
| INIT string type csvName as the name of the CSV file being loaded | 1 | 1 | 1 |
| INIT vector <Course Type> courses | 1 | 1 | 1 |
| WHILE choice is NOT 9 THEN | 1 | N | N, 1 |
| DISPLAY “Menu:” | 1 | 1 | 1 |
| DISPLAY NEWLINE | 1 | 1 | 1 |
| DISPLAY “1. Load Courses” | 1 | 1 | 1 |
| DISPLAY NEWLINE | 1 | 1 | 1 |
| DISPLAY “2. Print all Courses” | 1 | 1 | 1 |
| DISPLAY NEWLINE | 1 | 1 | 1 |
| DISPLAY “3. Find Course” | 1 | 1 | 1 |
| DISPLAY NEWLINE | 1 | 1 | 1 |
| ASSIGN choice with input from the user. | 1 | 1 | 1 |
| SWITCH (choice) | 1 | 1 | 1 |
| CASE 1 THEN | 1 | 1 | 1 |
| ASSIGN courses with CALL to loadVector with csvName | 1 | 1 | Best Case: 11  Worst Case: 10 + N(8 +N(2) + 4 + N(5) + 3) + 2 = 12 + 22N^2 |
| BREAK | 1 | 1 | 1 |
| CASE 2 THEN | 1 | 1 | 1 |
| INIT integer type coursesSize as the size of courses | 1 | 1 | 1 |
| INIT integer type index1 as 0 | 1 | 1 | 1 |
| INIT integer type index2 as 0 | 1 | 1 | 1 |
| WHILE index1 is less than coursesSize THEN | 1 | N | N, 1 |
| OUTPUT courses[index1].courseNumber | 1 | 1 | 1 |
| OUTPUT “, “ | 1 | 1 | 1 |
| OUTPUT courses[index1].courseName | 1 | 1 | 1 |
| INIT integer length as size of courses[index1].preReq | 1 | 1 | 1 |
| WHILE index2 is less than length THEN | 1 | N | N, 1 |
| OUTPUT “, “ | 1 | 1 | 1 |
| OUTPUT courses[index1].preReq[index2] | 1 | 1 | 1 |
| IF index2 does NOT equal (length minus 1) THEN | 1 | 1 | 1 |
| OUTPUT "," | 1 | 1 | 1 |
| INCREMENT index2 | 1 | 1 | 1 |
| END WHILE | 1 | 1 | 1 |
| INCREMENT index1 | 1 | 1 | 1 |
| END WHILE | 1 | 1 | 1 |
| BREAK | 1 | 1 | 1 |
| CASE 3 THEN | 1 | 1 | 1 |
| INIT string type courseNum | 1 | 1 | 1 |
| DISPLAY “Please input the Course Number you are searching for. | 1 | 1 | 1 |
| DISPLAY NEWLINE | 1 | 1 | 1 |
| INPUT user inputs a Course Number | 1 | 1 | 1 |
| ASSIGN courseNum with INPUT | 1 | 1 | 1 |
| CALL printCourse with courses, courseNum | 1 | 1 | Best Case: 8  Worst Case: 16 + 3N + 5N + 1 |
| BREAK | 1 | 1 | 1 |
| CASE 9 THEN | 1 | 1 | 1 |
| EXIT | 1 | 1 | 1 |
| END WHILE | 1 | 1 | 1 |
| **Total Cost** | | | Best Case: 3 + 16N  Worst Case: 3 + N(12 + 12 + 22N^2 + 1) = 3 + 47N^3 |
| **Runtime** | | | Best case: O(N)  Worst case: O(N^3) |