CL118 Programming Fundamentals Lab

Tuesday, June 23, 2020

Course Instructor

Tahir Farooq, Awais Azam, Ch. Usman Ghous

Serial No:

Final Term Exam

Spring Semester 2020

Max Time:3 Hour

Max Marks: 100

Exam Weight (Out of 100). 15

Roll No	Section

Guidelines for Submission:

1. You should submit only one PDF document and **all text should be typed**. Equations, figures can be taken as pictures (all figures/equations can be pasted as images inside that document).
2. You must submit your solution before due time via **Slate/Google Classroom**. Submissions submitted after the due time shall not be considered.
3. If you don't finish every part of a question, don't worry! You can still submit what you've done to get marks based on your efforts.
4. In case of copied or plagiarized solutions in exam Or If a student provided help to another student during exam both will be awarded "F" grade and it will affect the student CGPA.
5. Viva of any student can be conducted by the instructor after conducting an online exam in case of any doubt.
6. This document should be submitted through LMS (**Slate/Google Classroom)**. But in worst case, you can email it within the deadline.
7. Lastly, please make sure your submitted file has the naming convention RollNumber_PFLAB_FINAL

Question 1	**Question 2**	**Question 3**	**Total**
20	**20**	**60**	**100**

Question No. 1

FAST CFD campus is constructing a website for its faculty members. Currently the website is under construction, but the login module is completed. As the website is for faculty so only authorized faculty members can login. The emails of authorized faculty members are placed in a text file “facultyemails.txt”. Currently we have the following five emails in the text file:

1. tahir.farooq@nu.edu.pk
2. usman.ghous@nu.edu.pk
3. awais.azam@nu.edu.pk
4. shoaib.saleem@nu.edu.pk
5. m.husnain@nu.edu.pk

You are required to write a c++ program which asks for email and password. Currently the password is “123” for all emails. When the user enters email and password, your program should check if the entered email is present in text file “facultyemails.txt”. If the email is present in the text file it means faculty member can access the website and it should display a message “Website is Under Construction” otherwise if the email is not present in the text file it should display “You are not authorized to access this website”.

Question No. 2

You are required to design a software that is a voting system for FAST students. The voting system is to conduct a survey whether the university should continue its study operations by calling the students physically in university or not? Write a menu driven program that asks for email address, name, age and choice (y or n). “Y” means yes and “n” means no. Your program should not allow one email to be entered twice. And only allow the user to vote if the age is 18 or above. There should be an option in the menu to terminate the program and in the end the program should display the following:

1. Total number of votes received.
2. Votes entered using official NU email (email that contains @nu.edu.pk)
3. Votes entered using personal email (email that contains @gmail.com, @yahoo.com etc)
4. Total votes in favor of opening university.
5. Total votes opposing opening of university.

Question No. 3

One of the more interesting puzzlers for chess buffs is the Knight's Tour problem. The question is this: Can the chess piece called the knight move around an empty chessboard and touch each of the 64 squares once and only once? We study this intriguing problem in depth in this exercise.

The knight makes L-shaped moves (over two in one direction then over one in a perpendicular direction). Thus, from a square in the middle of an empty chessboard, the knight can make eight different moves (numbered 0 through 7) as shown in Figure below.

a) Draw an 8-by-8 chessboard on a sheet of paper and attempt a Knight's Tour by hand. Put a 1 in the first square you move to, a 2 in the second square, a 3 in the third, etc. Before starting the tour, estimate how far you think you'll get, remembering that a full tour consists of 64 moves. How far did you get? Was this close to your estimate?

b) Now let's develop a program that will move the knight around a chessboard. The board is represented by an 8-by-8 two-dimensional array board. Each of the squares is initialized to zero. We describe each of the eight possible moves in terms of both their horizontal and vertical components. For example, a move of type 0, as shown in Figure below, consists of moving two squares horizontally to the right and one square vertically upward. Move 2 consists of moving one square horizontally to the left and two squares vertically upward. Horizontal moves to the left and vertical moves upward are indicated with negative numbers. The eight moves may be described by two one-dimensional arrays, horizontal and vertical, as follows:

Horizontal [0] = 2	Vertical [0] = -1
Horizontal [1] = 1	Vertical [1] = -2
Horizontal [2] = -1	Vertical [2] = -2
Horizontal [3] = -2	Vertical [3] = -1
Horizontal [4] = -2	Vertical [4] = 1
Horizontal [5] = -1	Vertical [5] = 2
Horizontal [6] = 1	Vertical [6] = 2
Horizontal [7] = 2	Vertical [7] = 1

Let the variables currentRow and currentColumn indicate the row and column of the knight's current position. To make a move of type moveNumber, where moveNumber is between 0 and 7, your program uses the statements

```
currentRow += vertical[ moveNumber ];
currentColumn += horizontal[ moveNumber ];
```

Keep a counter that varies from 1 to 64. Record the latest count in each square the knight moves to. Remember to test each potential move to see if the knight has already visited that square, and, of course, test every potential move to make sure that the knight does not land off the chessboard. Now write a program to move the knight around the chessboard. Run the program. How many moves did the knight make?

c) After attempting to write and run a Knight’s Tour program, you’ve probably developed some valuable insights. We’ll use these to develop a heuristic (or strategy) for moving the knight. Heuristics do not guarantee success, but a carefully developed heuristic greatly improves the chance of success. You may have observed that the outer squares are more troublesome than the squares nearer the center of the board. In fact, the most troublesome, or inaccessible, squares are the four corners.

Intuition may suggest that you should attempt to move the knight to the most troublesome squares first and leave open those that are easiest to get to, so when the board gets congested near the end of the tour, there will be a greater chance of success.

We may develop an “accessibility heuristic” by classifying each square according to how accessible it’s then always moving the knight to the square (within the knight’s Lshaped moves, of course) that is most inaccessible. We label a two-dimensional array accessibility with numbers indicating from how many squares each particular square is accessible. On a blank chessboard, each center square is rated as 8, each corner square is rated as 2 and the other squares have accessibility numbers of 3, 4 or 6 as follows:

2 3 4 4 4 4 3 2
3 4 6 6 6 6 4 3
4 6 8 8 8 8 6 4
4 6 8 8 8 8 6 4
4 6 8 8 8 8 6 4
4 6 8 8 8 8 6 4
3 4 6 6 6 6 4 3
2 3 4 4 4 4 3 2

Now write a version of the Knight’s Tour program using the accessibility heuristic. At any time, the knight should move to the square with the lowest accessibility number. In case of a tie, the knight may move to any of the tied squares. Therefore, the tour may begin in any of the four corners. [*Note:* As the knight moves around the chessboard, your program should reduce the accessibility numbers as more and more squares become occupied. In this way, at any given time during the tour, each available square’s accessibility number will remain equal to precisely the number of squares from which that square may be reached.] Run this version of your program. Did you get a full tour? Now modify the program to run 64 tours, one starting from each square of the chessboard. How many full tours did you get?

d) Write a version of the Knight’s Tour program which, when encountering a tie between two or more squares, decides what square to choose by looking ahead to those squares reachable from the “tied” squares. Your program should move to the square for which the next move would arrive at a square with the lowest accessibility number.

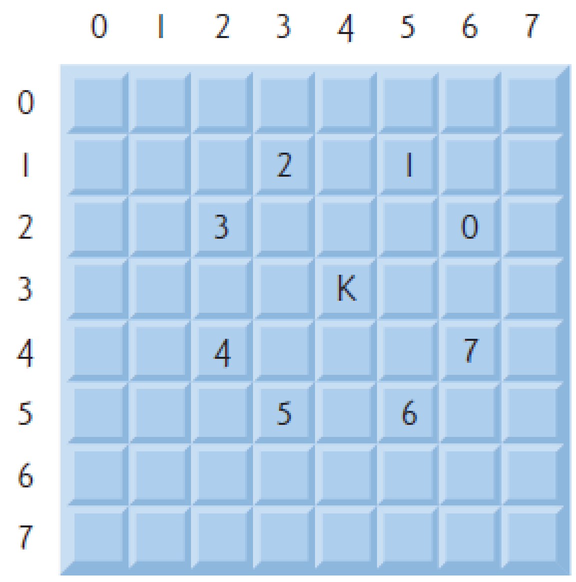

Figure 1: Eight possible moves of the Knight