1. PROBLEM DEFINITION

Using a python program to obtain data from a user of the tournament’s participants names and scores and organize them into a table for each of the golf team’s tournaments as well as determine the winner based of the scores put in.

1. ANALYSIS

I used loops to ensure proper inputs and prevent the program from giving unreasonable responses or blank inputs that would cause issue for the program later.

I chose to save scores as integers in order to compare them again 0 and 999 as I am unsure of the maximum scorable points in an 18 + hole game of golf, so using 999 as an upper barrier to ensure the input is a 3 digit or less number and 0 to ensure that all numbers are non-negative on input. Using an integer format while the scores are saved in variable ‘score’ also allows for easier formatting on the table.

Names are simply saved as strings as well as the ‘y/n’ inputs (‘inputAgain’ for beginning or leaving the input loop) for saving to the file and reading easily and to compare inputs to results accurately respectively.

Lists were used to store the names (‘namelist’) and scores(‘scorelist’) in order so that the index for one also referred to the other.

Start date (‘dStart’) and End Date (‘dEnd’) were stored as integers for comparison to ensure they were calendar days and in numeric format

YEAR and month were saved as strings so that long-form words or numbers could be used in either case, this does come at the cost of not having extra checks in place

Variable ‘formatlen’ was used to compare the length of the name versus the spacing allotted for the names and scores and align them properly with each other in a column, this is an integer

Variable ‘golfRecord’ was used to reference the file ‘golf.txt’ for all actions regarding it.

1. DESIGN
2. Define the main function to call inputInfo function and displayReport function in order
3. Inside the input info function
4. Ask user if they have any golf cards to input, if no exit function if yes proceed
5. Open ‘golf.txt’ for writing, erasing any previous golf.txt files
6. Ask user for the golfer’s name
7. Ask user for golfer’s score
8. Ensure score is not a negative and is not more than 3 digits
9. Write name to the file (new line for subsequent names)
10. Write the score to the file on the next line
11. Ask user if they have any more golfer’s cards to input, if yes repeat from step 3, if no close file and exit function
12. Inside displayReport function:
13. Open golf.txt for reading
14. Inititialize lists for name and score
15. Ask user for date of tournament
16. Create header for table using user’s input for the date
17. Read name from file store as ‘name’
18. Read score from file and store as ‘score’
19. Strip newline characters from name and file
20. Convert score to an integer
21. Add current ‘name’ to ‘namelist’
22. Add current score to ‘scorelist’
23. Calculate spacing based off length of the current ‘name’
24. Print name and score in the same line under their appropriate headers and send following name and score to next line
25. Read next name and store to ‘name’ if there is a name, loop from step 6, if there is no name (blank string returned) continue
26. Find the lowest score in ‘scorelist’ (all of the scores in the table) store as ‘minscore’
27. Find the index in the list of the lowest score and store as ‘index’
28. Print the winner in the form ‘Winner is (name of winner from ‘namelist’ with index of the lowest score, same as that from ‘scorelist’) with score ‘minscore’(the lowest score)
29. Close the file golfRecord (golf.txt)
30. Return to main function to end
31. IMPLEMENTATION

To program I used the IDLE environment on a x64 version of Windows 7 with an AMD FX-8120 processor. The data sets used varied between various inputs in the place of ‘y’ and ‘n’ including various strings of different lengths and their capital letter variants, for names I used my own name and adding or removing various characters from it and attempted other strings including special characters and numbers, all of which are accepted, and various strings from letters to numbers of varying length to test the numbers end and combined these for testing the layout of the table. The Program will not crash from any tested input, as most exceptions are handled and moved into a loop that will terminate when an appropriate input is reached (for integers values when asked, avoiding blank inputs to file, though month and year are left unguarded for freedom by the user which can still result in typos. Should strings be entered where an integer is expected the user will be told to enter a whole number there then loop to the input as many times as required.