Linked List Assignment

Duration: **3½ classes**
Due: **Beginning of class Thursday 13 December, 2012**

Survivor®:
Plan and write a structured program to simulate the TV show Survivor®. It will allow a user to enter an undetermined number of names of the show's contestants. The user will indicate that she has finished entering names by entering "fin" in the input prompt. The names will be stored as a *circular* linked list. (This is created by making the first element of the linked list point back to the last element.)

Back-Up:
Once the user has finished entering the list, the program will back up the *circular* linked list using a method. The need for this back-up will become apparent in the summary. (Note that the back-up should also be a *circular* linked list.)

Elimination:
After the back-up is complete, the user should be prompted to enter the names of the show's contestants to be eliminated, one-by-one, until there is only one contestant left (the winner). If the name that a user enters is missing from the list, an appropriate message should be displayed to the user before allowing another name to be entered.

Summary:
After the contestant elimination is complete, the program should use a recursive method to display a list of all the original contestants (using the back-up), followed by a congratulatory message to the winner.

Testing:
In addition to thoroughly testing your program, a further demonstration of your program must be performed: Enter the names of exactly 8 contestants. Eliminate each one, one-by-one.

(A listing of the output of this test will be submitted, as described below.)

Format:
This assignment is to be handed in as a hard copy. Provide only the program listing and a listing of the output of the test case described above.

Marking Scheme for Assignment:
Please refer to the attached rubric.

Rubric

Categories	Level 1 (50 – 59%)	Level 2 (60 – 60%)	Level 3 (70 – 79%)	Level 4 (80 – 100%)
Program Logic and Design				
Use of appropriate control structures (e.g. if-else, case, loops…)	inappropriate choice and use of control structures	occasional choice and use of appropriate control structures	frequent choice and use of appropriate control structures	consistent choice and use of appropriate control structures
Program easy to modify (e.g. use of const, types…)	modifications require extensive program changes	modifications require minor program changes	program relatively easy to modify	program easy to modify quickly and efficiently
Program Execution & Usage				
Successful program execution (normal and special input cases)	program runs partially or runs mostly incorrectly for normal input	program runs successfully for most normal input cases	program runs successfully and correctly for most normal input cases	program runs successfully and correctly for normal and special input cases
User friendliness (error, I/O messages)	apparent lack of I/O and error messages	occasional use of I/O and error messages	frequent use of I/O and error messages	consistent accurate use of I/O & error messages
Program Specifications				
Program meets assignment specifications (language features, I/O requirements)	program satisfies few assignment specifications	program satisfies some of the assignment specifications, but many still not met	program satisfies assignment specifications mostly correctly	program satisfies assignment specifications completely and correctly
Coding Style				
Logical arrangement of code structure (modules, type, var, const…)	apparent lack of logical and consistent organisation of code structure	consistent but inappropriate organisation of code structure	mostly consistent and logical organisation of code structure	consistent, logical organisation of code structure applied throughout
Proper code indentation techniques	code indentation inconsistent and inappropriate	consistent but inappropriate code indentation	mostly consistent, appropriate code indentation	complete, consistent, appropriate indentation
Appropriate identifier names (procs, vars…)	inappropriate names used, but consistently	descriptive names used inconsistently	descriptive names used relatively consistently	descriptive names used consistently throughout
Documentation				
Detailed program header (title, purpose, course code etc.)	inappropriate program header or no header used	program header apparent but limited in detail	accurate, somewhat detailed program header	accurate, thoroughly detailed program header
Program comments (vars, consts, types, program modules…)	no comments or very limited comments apparent	Some limited, inappropriate or inaccurate comments apparent	comments apparent throughout but lack in detail or accuracy	thorough, accurate comments used consistently throughout
Creativity				
User interface (design, colour, interactivity, user-interest, creativity)	minimal interactivity, crude interface design	somewhat interactive and/or creative interface design	interesting, interactive, user-friendly interface design	highly advanced, interactive, user-friendly interface design
Language features and concept complexity (stadard and advanced)	no advanced and lack of standard language features and concepts used	standard language features and concepts used correctly	some advanced language features and/or complex concepts used correctly	highly complex concepts and language features used correctly