Soccer Club

**Due Time:** 23.59, 29 January 2016 **Earnings:** 4% of your final grade

***NOTE: The code in this assignment must be your own work. It must not be code taken from another student or written for you by someone else, even if you give a reference to the person you got it from (attribution); if it is not entirely your own work it will be treated as plagiarism and given a fail mark, or less.***

**Purpose:** You are to write the code for a C language console application for a Soccer Club data base of players using a double-linked-list with dynamic memory allocation for its data. This will give you an opportunity to review material that has already been taught in earlier courses and get up to speed in C programming that will be needed for C++. Part of the code (minus the usual headers) is shown on the next page. You **MUST** use this code **without modification (not a single character changed): no code added or removed, no macros, no defines and no statics**. Your task is to implement only the functions that are declared and not add any new ones. All your code is in a single file named **ass0.c** (not ass0.cpp)

In this assignment, when the application is running the user can

* Add a new Player at the start of the list with name and subscription
* Delete the player at the start of the list
* Print all the players starting from the front of the list
* Print all the players starting from the end of the list

An example of the output of the running application is given at the end. Yours must look identical.

Note the following:

* The file must be named **ass0.c** (**not ass0.cpp**).
* The global g\_pPlayers holds the address of the first Link (the head) of the double-linked-list. When a new player is added a new Link for it is allocated in dynamic (heap) memory and inserted at the head of the list. The new Link has a pointer to the new Player in dynamic memory that has pointers to the first and last names of the player – these are also allocated on the heap in exactly as much memory as they need and no more.
* You must use functions like strlen() and strcpy() or similar etc. from the standard C library to handle strings. You cannot use the C++ string class.
* You cannot use any C++ constructs or classes. You must only use malloc() and free() (not realloc) for dynamic memory management.
* When the application terminates it releases **all** dynamically allocated memory (or you lose 30%).

An example of the output of the running application is given at the end. Yours must look identical.

See the Marking Sheet for how you can lose marks, but you will lose 60% if:

1. you change the supplied code in any way at all (not a single character) - no code added (except header includes) or removed, no macros, no defines, no statics and no additional functions,

2. it fails to build in Visual Studio 2013

3. It crashes in normal operation (such as printing or deleting from an empty list etc.)

4. it doesn’t work like the example output

Part of the code is shown on the next page. You MUST use this code **without modification.** Your task is to add the implementation of the functions that are declared using the style of the posted Submission Standard. All the code is in a single file named **ass0.c**.

**What to Submit :** Use Blackboard to submit this assignment as a zip file (**not** RAR) containing only the single source code file (ass0.c). If you are taking both CST8219 and CST8233, you need only to submit to CST8219. The name of the zipped folder **must** contain your name as a prefix so that I can identify it, for example, for CST8219, using my name the file would be tyleraAss0CST8219.zip. It is also vital that you include the Cover Information (as specified in the Submission Standard) as a file header in your source file so the file can be identified as yours. Use comment lines in the file to include the header. **Before you submit the code, check that it builds and executes in Visual Studio 2013 as you expect - if it doesn’t build for me, for whatever reason, you get a deduction of at least 60%**. There is a late penalty of 25% per day. Don’t send me the file as an email attachment – it will get 0.

***Example code: don’t change or add to it (not even a single character), but include the usual headers***

enum{ RUNNING = 1 };

struct Player

{

double subscription;

char\* firstName;

char\* lastName;

};

struct Link

{

struct Player\* pPlayer;

struct Link\* pNext; // Link to next object

struct Link\* pPrev; // Link to previous object

};

void AddPlayer();

void DeletePlayer();

void PrintPlayersForward();

void PrintPlayersReverse();

void CleanUp();

struct Link\* g\_pPlayers = 0;

int main()

{

char response ;

while(RUNNING)

{

printf("\nPlease select an option:\n");

printf("1. Add a Player\n");

printf("2. Delete a Player\n");

printf("3. Print out Players\n");

printf("4. Print out Players in reverse order\n");

printf("q. Quit\n");

printf("CHOICE: ");

fflush(stdin);

scanf("%c",&response);

switch(response)

{

case '1':AddPlayer();break;

case '2':DeletePlayer();break;

case '3':PrintPlayersForward();break;

case '4':PrintPlayersReverse();break;

case 'q':CleanUp();return 0;

default:printf("Please enter a valid option\n");

}

printf("\n");

}

}

***Example Output (yours must work identically)***

**Please select an option:**

**1. Add a Player**

**2. Delete a Player**

**3. Print out Players**

**4. Print out Players in reverse order**

**q. Quit**

**CHOICE: 1**

**ADDING AN PLAYER**

**Please enter the Player first name: Bill John**

**Please enter the Player last name: Smith**

**Please enter the Player subscription: 100.00**

**Please select an option:**

**1. Add a Player**

**2. Delete a Player**

**3. Print out Players**

**4. Print out Players in reverse order**

**q. Quit**

**CHOICE: 1**

**ADDING AN PLAYER**

**Please enter the Player first name: Mary Jo**

**Please enter the Player last name: Bell**

**Please enter the Player subscription: 120.00**

**Please select an option:**

**1. Add a Player**

**2. Delete a Player**

**3. Print out Players**

**4. Print out Players in reverse order**

**q. Quit**

**CHOICE: 3**

**FORWARD PRINTING PLAYERS**

**Player 1**

**first name = Mary Jo;**

**last name = Bell;**

**subscription = 120.00;**

**Player 2**

**first name = Bill John;**

**last name = Smith;**

**subscription = 100.00;**

**Please select an option:**

**1. Add a Player**

**2. Delete a Player**

**3. Print out Players**

**4. Print out Players in reverse order**

**q. Quit**

**CHOICE: 4**

**REVERSE PRINTING PLAYERS**

**Player 2**

**first name = Bill John;**

**last name = Smith;**

**subscription = 100.00;**

**Player 1**

**first name = Mary Jo;**

**last name = Bell;**

**subscription = 120.00;**

**Please select an option:**

**1. Add a Player**

**2. Delete a Player**

**3. Print out Players**

**4. Print out Players in reverse order**

**q. Quit**

**CHOICE: 2**

**DELETE A PLAYER**

**Player deleted**

**Please select an option:**

**1. Add a Player**

**2. Delete a Player**

**3. Print out Players**

**4. Print out Players in reverse order**

**q. Quit**

**CHOICE: 3**

**FORWARD PRINTING PLAYERS**

**Player 1**

**first name = Bill John;**

**last name = Smith;**

**subscription = 100.00;**

**Please select an option:**

**1. Add a Player**

**2. Delete a Player**

**3. Print out Players**

**4. Print out Players in reverse order**

**q. Quit**

**CHOICE: 4**

**REVERSE PRINTING PLAYERS**

**Player 1**

**first name = Bill John;**

**last name = Smith;**

**subscription = 100.00;**

**Please select an option:**

**1. Add a Player**

**2. Delete a Player**

**3. Print out Players**

**4. Print out Players in reverse order**

**q. Quit**

**CHOICE:**