CSE213L: DSA Lab
LNMIIT, Rupa Ki Nangal
Training Set 04

In this training set, students will be guided to write a simple program for a game which resembles a game of musical chairs. The primary data-structured used in the program will be a doubly linked list. For details of data-structure ADT interface functions and their implementation guidance, please refer to Section *6.5: Circular Doubly Linked List* in DSA textbook by R Thareja. Other, commonly used data structure textbooks are equally good for this purpose.

The interface we adapt is a little different from the one discussed in DSA class and the textbooks.

As has been instructed in the previous DSA Lab training sets, students will be encouraged to implement the data-structure and the application program in separate files: `dlist.c` and `main.c`. The interface functions and other macros will be declared in file `dlist.h`. This file is listed here to assist the novice student programmers.

```
#include <stdio.h>
#include <string.h>
#include <assert.h>
#include <stdlib.h>

struct chair {
      struct chair *leftChair;
      char *player;
      struct chair *rightChair;
};

/*    Free all memory that is accessible to dlist
      from pointer marking its start. This includes memory
      with struct as well as with string player
*/
void initDList();

/*    Create a new chair for a player. Add it to the
      list. Terminate prog if memory not available
*/
void insertChair(char *player);

/*    Free person name string and chair too.
      Count the fisrt pointed chair as numbered 0.
      Displacement can be positive or negative from
      chair 0. Player pointer should not be NULL.
*/
char *removeChair(int displacement);

/*    Return name of the person in the chair */
char *whoInChair(int displacement);

/* Is there any chair left in the game? */
int isEmpty();
```

Please see the end of this document for an output produced by an implementation of the game.

Training Set 04: Task 01

Create a text file `player` with names of 20 persons. Keep each name a distinct name. Type one name in each line of the file. These names will be read from the file and inserted into a circular list that you will write in this training set. As this is a game like a game of musical chairs, nodes in the list are tagged as `chair`. Some snippets of the code from file `main.c` are listed below.

Once the names have been read from file `players` into the list, the game can be played. The game will be played by rolling a dice. You already know from your childhood days that a roll of a dice returns a number from 1 to 6 with equal odds.

In each phase of this game, a dice will be rolled, and a player selected to be sacrificed and removed from the game. This player is called *victim*. The victim is selected as follows: The chair pointed to by the primary reference of the doubly linked list is chair 0. Rolled number denotes the victim's chair number. If the rolled value is an odd integer, the victim is chosen by counting the rolled value using the left links starting from chair 0. Otherwise, the right link is used to count chairs. The victim is removed from the list and victim's name printed on the standard output.

Carefully read file `main.c` below and write minimal codes for the ADT interface functions in file `dlink.c`. The purpose at this stage is to let the program compile correctly. The minimal implementation code needed in a function is usually the main block of the function with either no code or with a single return statement. For example,

```
char *whoInChair(int displacement) {
      return "None found";
}
```

It is quite common to add statement `printf("NOT IMPLEMENTED\n");` in these blocks.

Next, implement functions `initDList()` and `isEmpty()` after defining a global variable for the *starting* chair in program variable declarations. The starting chair is a member node in the doubly linked circular list of chairs from where the search for the victim begins. This is accomplished, in my program, by variable declaration (in file `dlink.c`):

```
struct chair *pointAtChair0;
```

A suggested code for file `main.c` is given below for the students to read and to let them use it to begin their training session.

```
#include <stdio.h>
#include <stdlib.h>
#include "dlink.h"

char tmp[101];

int rollDice() {
      int i = 1+rand()%6;
      return i;
}
```

```
int main(void) {
      char *name;
      int dice;

      initDList();
      FILE *playerF = fopen("players","r");
      assert(player != NULL); // File is open
      fgets(tmp, 100, playerF);
      while (!feof(playerF)) {
            // Get space filled with \0
            name = calloc(1,strlen(tmp)+1);
            // Newline should not copied
            strncpy(name, tmp, strlen(tmp)-1);
            insertChair(name);
            fgets(tmp, 100, playerF);
      }

      printf("\nAll palyers are in their chairs now.” “
                        Game begins\n\n");

      while (!isEmpty()) {
            printf("%s is in chair 0. ", whoInChair(0));
            dice = rollDice();
            if (dice%2 == 1) // Odd numbers on left!
            dice *= -1;
            printf("Rid chair %d. ", dice);
            name = removeChair(dice);
            if (isEmpty())
                  printf("\n\nWinner is %s\n", name);
            else
                  printf("\nLosing player was %s.\n", name);
            free(name);
      }
      return 0;
}
```

Once a student has set up the program and has correctly implemented and tested the two interface functions named above, the student may get this task marked by a lab tutor.

Training Set 04: Task 02

To complete Task 02, the student must implement all ADT interface functions except function `char *removeChair(int displacement);` This latter function is more difficult to implement and its implementation has been deferred till a later task (Task 03).

To claim that your interface functions in file `dlink.c` are working correctly, you may modify code in file `main.c` and use implemented ADT functions to list some player names from the doubly linked list.

Include `assert()` statements in our code to catch errors and avert mistakes. Program without the defensive programming checks may be vulnerable to unexpected failures and difficult to develop. DO NOT BE LAZY. Adapt error avoidance practices early.

Training Set 04: Task 03

Now is the time to implement a challenging task. Thask is implementation of function `char *removeChair(int displacement);` Study output of a game given below to identify different situations in which a chair could be removed from the list.

A helpful list of important situations is given below to the students. Keep solution code for each identified situation separate from solution codes for the other situations. Use `assert()` declarations to support this separation and to receive alert message if the separation is inadvertently violated.

A list of situations under which a chair may be removed from the linked list is suggested below. Take a careful note of these situations and how they impact the structure of the doubly linked list.

1. Removal of the last chair from the list
2. Removal of the second last chair from the list.
3. Removal (and identification) of the chair pointed to by the list's start pointer `pointAtChair0;`
4. Removal of other chairs.

The program output for a run of a game is listed below:

```
Adding Ram Mohan
Adding Sita
Adding Bharat Kumar
Adding John
Adding Charles
Adding Reeta
Adding Geeta
Adding Radha
Adding Najma
Adding Raman
Adding Mohammad
Adding Pooja
Adding Achla

All players are in their chairs now. Game begins


Ram Mohan is in chair 0 now.
        Rid chair 6.    Freeing a chair; at least 2 chairs left.
        Losing player was Geeta.
Ram Mohan is in chair 0 now.
        Rid chair 6.    Freeing a chair; at least 2 chairs left.
        Losing player was Radha.
Ram Mohan is in chair 0 now.
        Rid chair -5.   Freeing a chair; at least 2 chairs left.
        Losing player was Najma.
Ram Mohan is in chair 0 now.
        Rid chair -5.   Freeing a chair; at least 2 chairs left.
        Losing player was Reeta.
Ram Mohan is in chair 0 now.
        Rid chair 6.    Freeing a chair; at least 2 chairs left.
        Losing player was Mohammad.
Ram Mohan is in chair 0 now.
```

```
        Rid chair -5.   Freeing a chair; at least 2 chairs left.
        Losing player was John.
Ram Mohan is in chair 0 now.
        Rid chair -1.   Freeing a chair; at least 2 chairs left.
        Losing player was Achla.
Ram Mohan is in chair 0 now.
        Rid chair -1.   Freeing a chair; at least 2 chairs left.
        Losing player was Pooja.
Ram Mohan is in chair 0 now.
        Rid chair -5. Freeing chair 0.
        Losing player was Ram Mohan.
Raman is in chair 0 now.
        Rid chair -3.   Freeing a chair; at least 2 chairs left.
        Losing player was Sita.
Raman is in chair 0 now.
        Rid chair 6. Freeing chair 0.
        Losing player was Raman.
Bharat Kumar is in chair 0 now.
        Rid chair 6. Freeing second last chair.
        Losing player was Bharat Kumar.
Charles is in chair 0 now.
        Rid chair 2. Freeing the last chair.

Winner is Charles
```

Vishu Malhotra

30 April 2020
The LNMIIT, Rupa ki Nangal
Jaipur, Rajasthan 302031