“Musical Chairs”

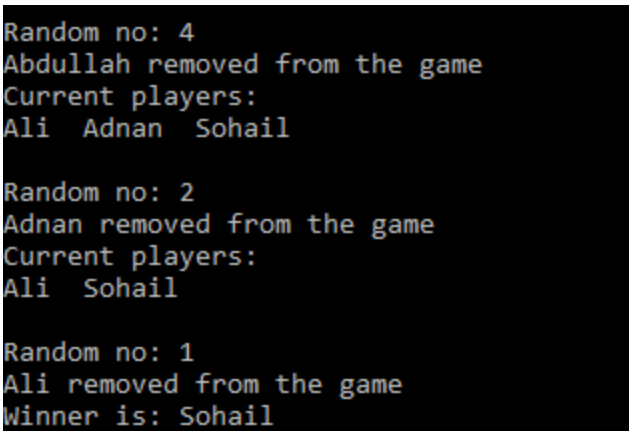
When you were a kid, you are might have played a game of musical chairs, where there are let’s say 5 children and 4 seats. The game involves children walking around the seats while music plays and trying to find a seat when the music stops. The child who fails to occupy a chair when the music stops is eliminated from the game, and then one seat is removed. This process continues until only one child remains as the winner.

- You will simulate this game using a single circular linked list data structure in C++. Each child participating in the game will be represented as a node in the single circular linked list. The linked list will be circular to simulate the continuous movement of children around the seats.
- Now actually you have to simulate this by creating 5 nodes initializing the list with the names of the players (hard coded using call to insert method).
- In the main menu you have two choices. Traverse the players one by one, and start game.
- **Traverse the players one by one:** Straight forwardly display the names in the list one by one, according to the user choice. See sample output.
- **Start game:** Start the game, and hen in each round you will generate a random no between 1 to current length of the list, and the node (player) at the position will be removed from the list. Note in each round, display the random no generated, player who is removed, and also display the current players in the game. In the end show the winner.

```
1. Traverse players one by one
2. Start game
3. Exit
1
Ali
Show:
1. Next
2. Go to main menu
1
Amir
Show:
1. Next
2. Go to main menu
1
Adnan
Show:
1. Next
2. Go to main menu
1
Sohail
Show:
1. Next
2. Go to main menu
1
Abdullah
Show:
1. Next
2. Go to main menu
1
Ali
Show:
1. Next
2. Go to main menu
2
1. Traverse players one by one
2. Start game
3. Exit
2
Current players:
Ali  Amir  Adnan  Sohail  Abdullah

Random no: 2
Amir removed from the game
Current players:
Ali  Adnan  Sohail  Abdullah

Random no: 4
Abdullah removed from the game
Current players:
Ali  Adnan  Sohail
```



“Cave Escape”

As a cave explorer you are going to enter in a dark cave, design a stack-based escape plan for an adventurer. The adventurer enters the cave and moves forward. He uses a stack to mark his path as he explores the cave. He must push the current location in the stack before moving forward, so that when he sees a ghost, he can go back to these points so that he does not get lost and find their way back to the exit. Implement stack using linked list.

```
Welcome to the cave
Must put your location info in a stack so that you can go back whenever you see a ghost
Enter a choice:
1. Go forward in the cave
2. Go back
3. Run from the cave.... I saw the ghost
1
Mark your current location, so that you can go back
Enter the name of the current location:
A
Enter a choice:
1. Go forward in the cave
2. Go back
3. Run from the cave.... I saw the ghost
1
Mark your current location, so that you can go back
Enter the name of the current location:
B
Enter a choice:
1. Go forward in the cave
2. Go back
3. Run from the cave.... I saw the ghost
1
Mark your current location, so that you can go back
Enter the name of the current location:
C
Enter a choice:
1. Go forward in the cave
2. Go back
3. Run from the cave.... I saw the ghost
2
You are now at: C
Enter a choice:
1. Go forward in the cave
2. Go back
3. Run from the cave.... I saw the ghost
1
Mark your current location, so that you can go back
Enter the name of the current location:
D
Enter a choice:
1. Go forward in the cave
2. Go back
3. Run from the cave.... I saw the ghost
3
You are now at: D
You are now at: B
You are now at: A
You successfully ran away from the cave
```

```
Welcome to the cave
Must put your location info in a stack so that you can go back whenever you see a ghost
Enter a choice:
1. Go forward in the cave
2. Go back
3. Run from the cave.... I saw the ghost
2
You have not even entered the cave
```