Problem 1:

Approach:

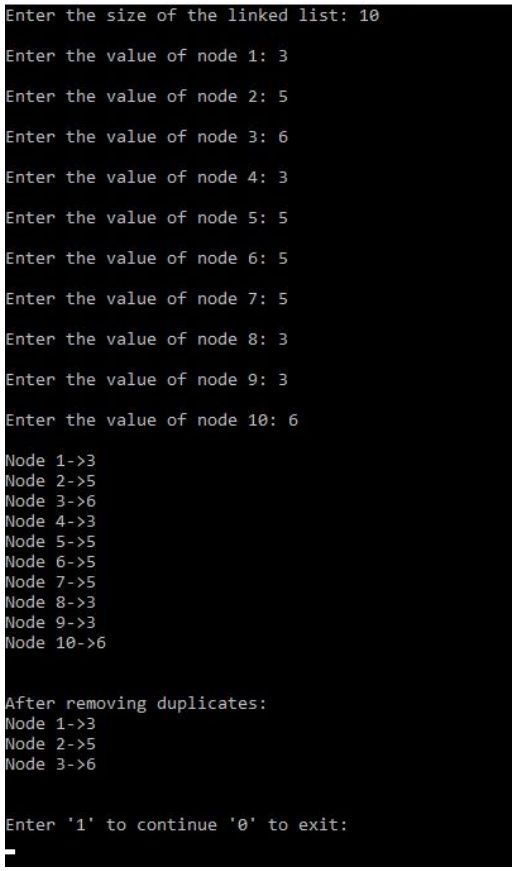
- Took user input for the size of linked list, and added the node values entered by the user.
- Declared two pointers, initially pointing to the head of the node.
- Used two while loops to compare the node values and remove the duplicate ones.
- Displayed the updated linked list with no duplicate values.

Why I used mutex in the addNode function?
If two or more users want to add a new node, then the first user to add a node will set a lock, i.e. it will not allow any other user to add a node, till the first user finishes adding the nodes. After which it will unlock the mutex.

Solution:
Open "Solution/Problem_1/bin/Debug/linkedList.exe" to see the executable for Problem 1.

Code files: main.cpp, linkedList.cpp, linkedList.h

Output:



Problem 2:

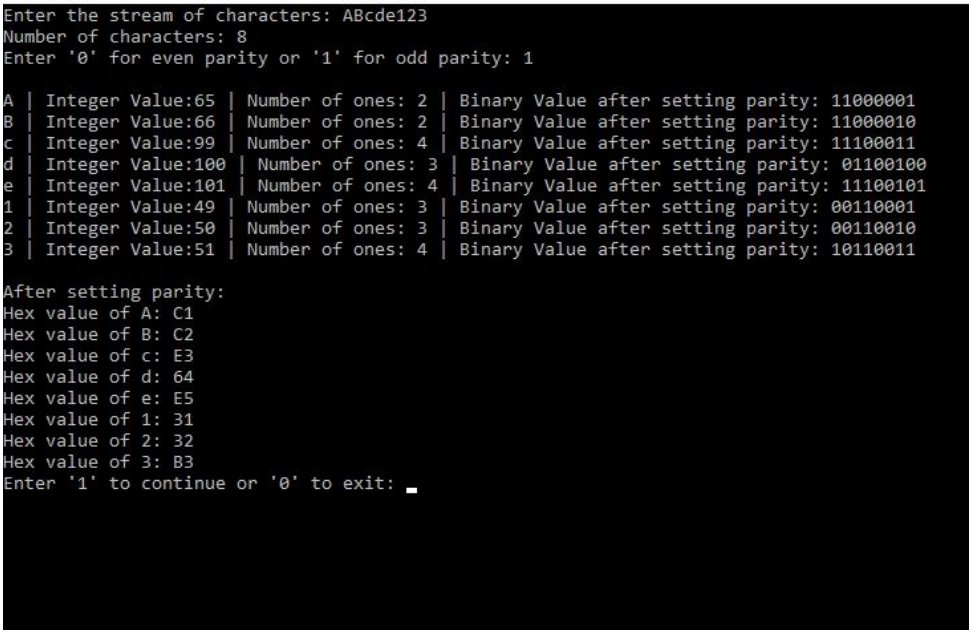
Approach:

- Took user input in the form of character array. (Assumption: Space is not considered as a character in this solution)
- Added an option where the user can select the setting to even or odd parity bit.
- Converted each character to its ASCII value.
- Counted the number of ones' present in the binary value of each of them and simultaneously added the parity bit as required, which makes it 8-bits.
- Divided the 8-bit value into two equal parts, and found the hexadecimal value for each part by converting them to integers.
- If the integer is between 0-9, added 48 to them, as the ASCII value is 48-57 respectively, and if the integer is between 10-15, added 55 to them, as the ASCII value is 65-70 respectively.
- Displayed the result in the form of characters of their respective ASCII value.

Solution:
Open "Solution/Problem_2/bin/Debug/two.exe" to see the executable for Problem 2.

Code files: main.cpp

Output:
For odd parity:



For even parity: