Lab 4 – C++ Stack & Queue

With the following struct:

```
struct node {
	int data;
	node *next;
};

struct list {
	node *pHead;
	node *next;
};

struct stack {
	node *top;
	int count;
};

struct queue{
	node *front;
	node *rear;
	int count;
};
```

Solve problem from 1 to 5. Or you can use self-define data structure.

Problem 1 *

Implement a stack along with some basic operations

a) Create: Creates an empty linked stack.
b) Push: Pushes new data into a stack.
c) Pop: Pops an element from the top of a stack.
d) Top: Retrieves data on the top of a stack without changing the stack.
e) isEmpty: Determines if a stack is empty.
f) isFull: Determines if a stack is full.
g) Clear: Clear a stack to make it empty.
h) Size: Determines the current number of elements in a stack.

Problem 2 *

Implement a queue along with some basic operations:

a) Create: Creates an empty linked queue.
b) EnQueue: Inserts one element at the rear of a queue.
c) DeQueue: Deletes one element at the front of a queue.
d) QueueFront: Retrieves data at the front of a queue without changing the queue.
e) QueueRear: Retrieves data at the rear of a queue without changing the queue.
f) isEmpty: Determines if a queue is empty.
g) isFull: Determines if a queue is full.
h) Clear: Clear a queue to make it empty.
i) Size: Determines the current number of elements in a queue.

Problem 3

a) Write a function that reverses a stack.
b) Write a function that reverses a queue.
c) Write a function that reverses a list by just using a stack and operations on that stack.
d) Write a function that reverses a list by just using a queue and operations on that queue.

Problem 4

Write a function that converts decimal radix into binary radix by using stack(s). User inputs a positive integer number N and the function will print out the binary radix of N on the screen.

For example:

```
// Problem 4
DecimalToBinary(10); // Print out: 1010
```

Problem 5

a) Write a function called `stackToQueue()` that creates a queue from a stack. After the queue has been created, the top of the stack should be the front of the queue and the base of the stack should be the rear of the queue. At the end of the algorithm, the stack should be empty.

b) Write a function called `queueToStack()` that creates a stack from a queue. At the end of the algorithm, the queue should be unchanged; the front of the queue should be the top of the stack, and the rear of the queue should be the base of the stack