Zig Zag Traversal

```c
#include <stdio.h>
#include <stdlib.h>
#define MAX 500

struct node
{
	int data;
	struct node *left;
	struct node *right;
};

struct node *newNode(int data)
{
	struct node *temp = (struct node *)malloc(sizeof(struct node));
	temp->data = data;
	temp->left = temp->right = NULL;
	return temp;
}

struct node **createStack(int *top)
{
	struct node **stack = (struct node **)malloc(sizeof(struct node *)*MAX);
	*top = -1;
	return stack;
}

void push(struct node **stack, struct node *node, int *top)
{
	stack[++(*top)] = node;
}

struct node *pop(struct node **stack, int *top)
{
	return stack[(*top)--];
}

int isStackEmpty(int *top)
{
	return (*top == -1);
}

void ZigZagTraversal(struct node *root)
{
	int top1, top2, top;
	struct node **stack1 = createStack(&top1);
	struct node **stack2 = createStack(&top2);
	struct node *temp;
	push(stack1, root, &top1);
	while(!isStackEmpty(&top1) || !isStackEmpty(&top2))
	{
		while(!isStackEmpty(&top1))
		{
			temp = pop(stack1, &top1);
			printf("%d\t", temp->data);
			if(temp->right)
				push(stack2, temp->right, &top2);
			if(temp->left)
				push(stack2, temp->left, &top2);
		}
		while(!isStackEmpty(&top2))
		{
			temp = pop(stack2, &top2);
			printf("%d\t", temp->data);
			if(temp->left)
				push(stack1, temp->left, &top1);
			if(temp->right)
				push(stack1, temp->right, &top1);
		}
	}
}


int main()
{
	struct node *root=NULL;
	root = newNode(10);
	root->left = newNode(20);
	root->right = newNode(30);
	root->left->left = newNode(40);
	root->right->left = newNode(50);
	root->right->right = newNode(60);
	ZigZagTraversal(root);
	return 0;
}
```

Time complexity is O(n)
Space complexity is O(n)