CS 2123
Data Structures
Midterm

NAME:_____________________________________

Instructions
1. Do all of the 4 problems
3. You have 50 minutes for the exam
4. Show all your work
5. Do not separate midterm papers
1. (25 pts) Assuming that temp1 points to a list of nodes that looks like this:

Write code that will insert a new node containing the value 7 into the list after the node containing 6 and before the node containing 12. You do not need to write code that searches for the values 6 or 12. Just use the pointer values shown in the picture.

List node definition is as follows

```c
struct node
{
    int info;
    struct node *next;
};
typedef struct node node;
```

Add your code below. Feel free to declare more variables as needed.

```c
node *temp1;
// code deleted that builds the list as shown above
....
// your code here
```
2. (25 pts) Write a function which takes two singly linked lists list1 and list2 and concatenates them. A pointer to the final concatenated list is returned by the function.

List node definition is as follows

```c
struct node
{
    int info;
    struct node *next;
};
typedef struct node node;
```

Function prototype is given below.

```c
node *concatenate(node *list1, node *list2)
{
    // Your code here
}
```
3. (25 pts) Explain how to implement a queue using two stacks. Think about the two functions enqueue(insert) and dequeue(delete) in a queue and explain how these functions can be implemented using a stack. Explain it using the following queue example

insert 1
insert 2
insert 3
delete 1
delete 2
insert 4
insert 5
delete 3
4. (25 pts) Below are the APIs for a queue and a stack of integers

Queue
    int empty(queue q)          returns 1 if queue is empty, 0 otherwise
    int delete(queue q)         deletes one int from queue and returns it
    void insert(queue q, int e)  inserts e into the queue

Stack
    int empty(stack s)          returns 1 if stack is empty, 0 otherwise
    int pop(stack *s)           pops top element from stack and returns it
    void push(stack *s, int e)   pushes e into stack

Using the functions of stack and queue, show how to reverse the items in a queue by using an intermediate stack.

stack *s;
queue q;
// items are inserted into queue here
// stack is initialized here