CS 2123
Data Structures
Midterm

---

### Question Points

<table>
<thead>
<tr>
<th>Question</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question 1</td>
<td></td>
</tr>
<tr>
<td>Question 2</td>
<td></td>
</tr>
<tr>
<td>Question 3</td>
<td></td>
</tr>
<tr>
<td>Question 4</td>
<td></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
</tr>
</tbody>
</table>

---

**NAME:**

---

**Instructions**

1. Do all of the 4 problems
2. You have 50 minutes for the exam
3. Show all your work
4. Do not separate midterm papers

---

<table>
<thead>
<tr>
<th>Easy</th>
<th>Difficulty Level</th>
<th>Difficult</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>
1. (25 pts) Assuming that temp1 points to a list of nodes that looks like this:

```
null ← 5 → 6 → 12 → null
```

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
```

Solution:

```c
node *p;
node *q;

p = temp1->next;
q = (node *)malloc(sizeof(node));
q->info = 7;
q->next = p->next;
p->next = q;
```
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.

**Solution:**

```c
node *concatenate(node *list1, node *list2)
{
    node *p;
    if (list1 == NULL)
    {
        list1 = list2;
        return (list1);
    }
    if (list2 == NULL)
        return (list1);
    p = list1;
    while (p->next != NULL)
    {
        p = p->next;
        p->next = list2;
    }
    return (list1);
}
```
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
```

**Solution:**

Have two stacks inbox and outbox.

**Insert:** Push the element onto inbox

**Delete** If outbox is empty, refill it by popping elements from inbox and pushing them onto outbox. Pop and return the top element from outbox.
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

Solution:

int x;
while (empty(q) == 0)
{
    x = delete(q);
    push(s,x);
}

while (empty(s) == 0)
{
    x = pop(s);
    insert(q,x);
}