Data Structures - Test-3

Name:_____

Question 1: Do 1.1-1.5

1.1 A queue is used to reorder an array of integers. Describe the ordering in the integer array `arr` after executing function `f()`?

```
template <typename T>
void f(T arr[], int n){
  queue<T> q;
  int i;

  for (i = 0; i < n; i++)
    q.push(arr[i]);

  i = 0;
  while (!q.empty()){
    arr[i] = q.front();
    q.pop();
    i++;
  }
}
```

(a) creates an ordered array in ascending order.

(b) creates an ordered array in descending order.

(c) reverses the order of the elements.

(d) maintains the same order.

1.2 Consider the Radix sort algorithm. The integer array `intArr` initially contains the following sequence of elements.

```
intArr:  {767, 41,  5, 212,  87,  543,  717, 18}.
```

(a) Show the 10 queues based on the units position and give the ordering of intArr after distributing elements based on the units position and collecting them back into the array.

(b) Show the 10 queues based on the tens position after (a) and give the ordering of intArr after collecting the elements back into the array.

1.3 Consider the `node` class defined below:

```
template <typename T>
class node
{
   public:
      T nodeValue;
      node<T> *next;

      node() : next(NULL)
      {}

      node(const T& item, node<T> *nextNode = NULL) :
          nodeValue(item), next(nextNode)
      {}
};
```

Use `node<char>` pointers for this question:

```
node<char> *p, *q, *r, *s, *newNode;
```

(a) After executing the instructions, display the order of the nodes.

```
p = new node<char>('M');
q = new node<char>('T');
r = new node<char>('B', p);
s = new node<char>('W', q);
p->next = s;
```

(b) Assume the order of the nodes from part (a). Give the code that would delete the node with value 'W' from the list.

(c) Give the code segment that would allocate `newNode` with value `'G'` and insert the node after node `r`.

1.4 Consider the class `tnode` defined below:

```
template <typename T>
class tnode
{
public:
  T nodeValue;
  tnode<T> *left, *right;

  tnode(){}

  tnode (const T& item, tnode<T> *lptr = NULL, tnode<T> *rptr = NULL):
    nodeValue(item), left(lptr), right(rptr)
  {}
};
```

Display the tree `root` created by the following statements.

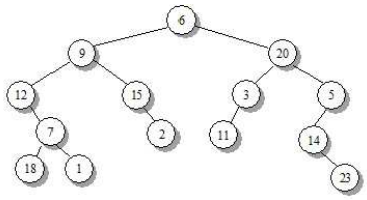
```
tnode<int> *root, *a, *b, *c, *d;

d = new tnode<int> (6);
c = new tnode<int> (9,d,NULL);
b = new tnode<int> (4,NULL,c);
a = new tnode<int> (12);
root = new tnode<int> (10,a, b);
```

1.5 Describe the action of `ct2`, and give the output on the tree shown below.

```
template <typename T>
int ct2 (tnode<T> *t)
{
  int ctLeft, ctRight, ct;

  if (t == NULL)
    return 0;
  else {
    return ct2(t->left)+ct2(t->right)+
           ((t->left != NULL && t->right != NULL) ? 1 : 0);
  }
}
```



Question 2

Write the following functions on the type `node` (see above for the definition).

(a) `occurs(p,x)` returns the number of occurrences of the element `x` in a list with its header pointed to by `p`.

```
template <typename T>
int occurs(node<T> *p, const T& x);
```

(b) `reverse(p)` reverses the linked list.

```
template <typename T>
node<T> *reverse(node<T> *p);
```

Question 3

Write the following functions on the type `tnode` (see above for the definition).

(a) Define the function `count` that returns the number of nodes in a binary tree.

```
template <typename T>
int count(tnode<T> *root);
```

(b) Define the function `leaves` that returns through the vector `v` the values of the leaves from left to right in a binary tree.

```
template <typename T>
void leaves(tnode<T> *root, vector<T>& v);
```