Data structures HW #5

(2) (a) 6, 3, 5, 4, 2, 3

(b) 2 1 1

(3)
```
bool compareStacks()
{ if( stack1.isEmpty() && stack2.isEmpty())
    { return true; }
  if( stack1.isEmpty() || stack2.isEmpty())
    { return false; }
  int n1= stack.pop(), n2= stack2.pop();
  bool same= (n1==n2);
  if (same)
  { same= compareStacks();
  }
  stack1.push(n1);
  stack2.push(n2);
  return same;
}
```

(1)
```
int function ( arr )
{ n= lengtharr
  int amount=0, place=1, n
return positivenumbers (arr, n, place, amount)
}
```

```
int positivenumbers (arr, n, place, amount)
{ if ( place > n)
    return amount;
  if ( arr[place] > 0)
    amount++
  return (positivenumbers (arr, n, place+1, amount))
}
```

(4) (a)
```
int quotient (num1, num2)
{ if (num1<num2)
    return 0;
  return (1+ (quotient (num1 - num2, num2)))
}
```

(b)
```
int modulus (num1, num2)
{ if(num1<num2)
    return num1;
  return (modulus (num1-num2, num2))
}
```

5 a

1. merge sort $O(n \log n)$
2. quick sort $O(n^2)$
3. merge $O(n+m)$

$$O(n \log n) + O(n^2) + (n+m)$$

worst case run time complexity in this algorithm is $O(n^2)$

b

quick sort $\Theta(n \log(n))$

merge sort $\Theta(n \log(n))$

merge $O(n)$

$$O(n) + \Theta(n \log(n)) + \Theta(n \log(n))$$