1. How many multiplications are being performed in the following code? Show your complete work. An answer alone is not sufficient to earn full credit. [8 pts]

```cpp
for (k = 11; k <= 20; k++) {
    for (i = 1; i <= k; i++) {
        w = t * k;
        p = z * i;
        for (j = 1; j <= 14; j++)
            s = k * t * j;
    }
}
```

\[
\sum_{k=11}^{20} \sum_{i=1}^{14} \left( 2 + \sum_{j=1}^{14} 2 \right)
\]

\[
\sum_{k=11}^{20} 30k
\]

\[
\left( \sum_{k=1}^{20} 30k - \sum_{k=1}^{10} 30k \right)
\]

\[
= 30 (20) \frac{21}{2} - 30 (10) \frac{11}{2}
\]

\[
= 4650
\]
2. Using summations find the value of count in terms of p after the following segment has been executed. Note that p is an even integer. Show your complete work. An answer alone is not sufficient to earn full credit [10 pts]

```c
count = 0;
for (i = 0; i < 16 * p + 16; i++) {
    for (j = p/2; j <= p; j++) {
        count++ = p - j;
    }
}
```

\[ M = 16p + 16 \]

\[ \sum_{i=0}^{M} \sum_{j=p/2}^{p} (p - j) \] \[2 \text{ pts}\]

\[ \sum_{i=0}^{M} \left[ \sum_{j=p/2}^{p} (p - j) - \sum_{j=p/2}^{p} j \right] \]

\[ \sum_{i=0}^{M} \left[ p(p-p/2+1) - \left( \sum_{j=1}^{p} j - \sum_{j=1}^{p/2-1} j \right) \right] \] \[4 \text{ pts}\]

\[ \sum_{i=0}^{M} \left[ p(1+p/2) - p(p+1)/2 + (p/2 - 1)(p/2) / 2 \right] \] \[2 \text{ pts}\]

\[ \sum_{i=0}^{M} \frac{p(p+2)}{8} \]

\[ = 16(p+1)p \frac{(p+2)}{8} \]

\[ = 2p(p+1)(p+2) \] \[2 \text{ pts}\]
3. Write a recursive function `struct node * largest( struct node * B)` which returns a pointer to the node containing the largest element in a BST (binary search tree). The node structure is as follows:

```
struct node {
    int node_value;
    struct node * left, *right;
};
```

```
struct node* largest(struct node *B){
    if ( B==NULL)
        return NULL;
    else if (B->right ==NULL)
        return B;
    else return largest(B->right);
}
```

Grading: 4 pts for an iterative solution

4. In a binary tree, each node may have a single child, two children, or no child. Write a recursive function `int one (struct tree_node *p)` for a binary tree which returns the number of nodes with a single child. Use the node structure

```
struct tree_node {
    int data;
    struct tree_node * left, *right;
};
```

```
int one ( struct tree_node *p){
    if ( p != NULL)
        return one(p)
    ...}
```


Solution 2:

int one (struct tree_node *p){
    if (p != NULL)
    {
        if( p->left == NULL && p->right != NULL
            || p->right == NULL && p->left != NULL)
            return 1+ one(p->left)+ one(p->right);
        else
            return (one (p->left) + one(p->right)) ;
    }
    return 0;
}
5. The following code is applied on the tree shown below with p pointing to the root of
the tree. Show each change on the tree by crossing out the old value and replacing with
the new value. [14 pts]

```c
struct node {
    int data;
    struct node * left, *right;
};
func( struct node *p)
{
    if ( p == null)
        return;
    func(p ->right);
    func( p->left);
    if (p->right != null)
        p ->data = p ->right -> data;
    if (p->left != null)
        p ->data = (p ->left -> data)/2;
}
```

Grading : One point for each change in value