Answer Pages

Question 21 (`pushAll`) answer:

~~void~~ ~~pushAll~~

```
void    Tree::pushAll ( TreeNode * n){
            if (n == NULL){    // don't need modify tree, just push data
                  return;}

            // push and call recursion
            st.push (temp);
            pushAll ( n-> leftChild );}
```

Question 22 (`KStep`) answer:

```
Tree:: kStep( ) {
        pushAll (root);
    while (st. isEmpty() != true) {
                pushAll( st. pop() -> right ); }
```

Question 23 (hasMore) answer:

```
bool Tree::hasMore() {
  treeNode * temp = root;
  if(root != NULL) {
    while(temp -> right != NULL) {
        temp = temp -> right; }
    if(temp -> data == st -> data) {
        return true; }}
  if( root == NULL) {
      return true; }
  return false; }
```

Question 24 (step1) answer:

```
int Tree::step1();
    treeNode * temp = st.pop();
    int out = temp -> data;
    return out; }
```

implementation of step(k);

Question 25 (~~step1 running time~~) answer:

```
int   Tree :: Step (int  k) {
      treeNode *  temp = st.top();
      for (int i=0, i<k, i++) {
            if (st.isEmpty()!=true) {
                  temp= st.pop(); }
            return ; }}
      if ( st.isEmpty()!=true) {
            int  out= st.top->data;
            return  out;}
      return
```

Question 26 answer:

Lower Bound	O(1)
Average	O(logn)
Upper Bound Case	O(n)

Question 27 (buildPerfectTree) answer:

```
QuadTreeNode *   QuadTree::buidPerfectTree(int k, RGBAPixel p){
        QuadtreeNode * temp = new QuadtreeNode();
        temp = buildHelper(0,0, pow(2,k), p, root);
        return temp;}
int   QuadTree:(int num, it, power){
      int out=1;
      for(int i=0; i<=k; i++){
           out=out*num;}
      return out;}
QuadTreeNode* QuadTree::buildHelper(int x, int y, int size, RGBAPixel p, QuadTreeNode curr){
      clear(root);
      if(resolution==1){
           curr = new QuadtreeNode();
           curr -> element = p;}
      buildHelper(x,y, size/2, p, curr->nwChild);
      buildHelper(x+size/2, y, size/2, p, curr->neChild);
      buildHelper(x, y+size/2, p, curr -> swChild);
      buildHelper(x+size/2, y+size/2, p, curr-> seChild);
```

Question 28 (perfectify) answer:

```
void   Quadtree::perfectify(int levels){
           int height = getHeight(root);
           perfectify(level - height, root);
int    Quadtree::getheight(Quadtree Node * curr){
           int i=0.
           if(curr == NULL){
                return i;}
           i = 1+ getheight(curr->nwChild);}
void   Quadtree::perfectiaty(int num, QuadtreeNode *& curr){
             if(curr -> nwChild == NULL){
             curr->nwChild = buildhelper(num, p);
             curr->neChild = buildhelper(num,p);
             curr->swChild = buildhelper(num,p);
             curr->seChild = buildhelper(num,p);}
             perfectiaty(num, curr->nwChild);
             perfectiaty(num, curr->neChild);
```

Question 29 (perfectify running time) answer:

```
             perfectiaty(num, curr->swChild);
             perfectiaty(num, curr->seChild);
```

Question 30 answer:

You may answer this question by filling in these blanks, or use the blank space for your own proof/disproof.

Preliminaries Let $H(n)$ denote the maximum height of an n-node SAVL tree, and let $N(h)$ denote the minimum number of nodes in an SAVL tree of height h. To prove (or disprove!) that $H(n) = \mathcal{O}(\log n)$, we attempt to argue that

$$H(n) \leq 3 \log_2 n, \text{ for all } n$$

Rather than prove this directly, we'll show equivalently that

a) $N(h) \geq$ $2^{(H(n)/3)}$, (1pt)

Proof For an arbitrary value of h, the following recurrence holds for all SAVL Trees:

b) $N(h) =$ 1 + $N(h-1)$ + $N(h-3)$, (2pt)

c) and $N(0) =$ 1, $N(1) =$ 2, $N(2) =$ 3, (2pt)

We can simplify this expression to the following inequality, which is a function of $N(h-3)$:

d) $N(h) \geq$ 2 $\times$ $N(h-3)$, (1pt)

By an inductive hypothesis, which states:

e) $N(h) \geq 2^{h/3}$, $N(0) \geq 2^{0}$, $N(1) \geq 2^{1/3}$, $N(2) \geq 2^{2/3}$, (1pt)

we now have

f) $N(h) \geq$ $2^{h/3}$ = part (a) answer, (1pt)

which is what we wanted to show.
Given that $2^0 = 1, 2^{1/3} \approx 1.25$, and $2^{2/3} \approx 1.58$, what is your conclusion?
Is an SAVL tree $\mathcal{O}(\log n)$ or not? (Circle one): (2pt)

YES NO

Overflow Page

Use this space if you need more room for your answers.