자료구조 과제(HW5)

전공: 철학과 학년: 3학년 학번: 20180032 이름: 남기동

1. Max Heap represented as a linked binary tree

<Pseudo Code>

Global Variable: treePointer root(the root of binary tree representing max heap), int number(number of nodes in tree), int stack[MAX_SIZE](saving

Local Variable: command(i means insert, d means delete, q means quit), result(1 means exist number)

Function:

1. void push (int command)

```
        if top == MAX_SIZE − 1, return error
        stack[++top] = command
```

2. int pop ( )

```
        if top == -1, return error
        return stack[top--]
```

3. void Insert (int key)

```
If( number == MAX_SIZE -1) printf("The heap is full")
temp = new node, temp->key = key, temp->leftchild = NULL
temp->rightchild = NULL; I = ++number;
```

```
if (i==1) // no node in tree
        root = temp; return;
while (i != 1)
        if ( i %2 == 0 ) push ( -1 ) // it means leftchild
        else push ( 1 ) // it means rightchild
        I /= 2;
ptr = root;
While (top != -1 ) {
        If( top == 0 )
                command = pop()
                If( command == -1)
                        If( temp->key > ptr->key ) Swap key
                        ptr->leftchild = temp; temp->parent = ptr;
                Else if(command == 1)
                        If( temp->key > ptr->key ) Swap key
                        ptr->rightchild = temp; temp->parent = ptr;
        else
                command = pop()
                If( command == -1)
                        If( temp->key > ptr->key ) Swap key
                        ptr = ptr->leftchild;
                Else if(command == 1)
```

```
                    If( temp->key > ptr->key ) Swap key
                    ptr = ptr->rightchild;
}
```

4. void Delete ( )

```
If(number == 0) printf("the heap is empty")
I = number
If( only one node in tree )
        Temp = root, free(temp), root = NULL, number--, printf("Delete %d", item)
While( i != 1 )
        if( I % 2 == 0) push(-1) //means leftchild
        else push(1) //means rightchild
        i /= 2
a = root, b = a->leftchild, c = a->rightchild;
while( top != -1) {
        if( top == 0)
                command = pop()
                if( command − 1)

        if ( b->key > c->key) a->key = b->key
        else if(b->key < c->key) a->key = c->key
        command = pop()
```

```
        if(command == -1 && b->key > c->key)
                a = a->leftchild, b = a->leftchild, c = a->rightchild
        if(command == -1 && b->key < c->key)
                c->key = b->key, (if c's child is bigger, then swap)
                a = a->leftchild, b = a->leftchild, c = a->rightchild
        if(command == 1 && b->key > c->key)
                b->key = c->key, (if b's child is bigger, then swap)
                a=a->rightchild, b = a->leftchild, c = a->rightchild
        if(command == 1 && b->key < c->key)
                a = a->rightchild, b = a->leftchild, c = a->rightchild
}
number--, printf("Delete %d", item)
```

5. void Search (treePointer, ptr, int key, int* result)

```
If( !ptr) return
If( ptr->key == key) *result = 1
Search(ptr->leftchild, key, &(*result))
Search(ptr->rightchild, key, &(*result))
```

6. main function

```
For ( ; ; ) {
        Get one character as command
```

Switch(command)

Case ‘I’

Get number for input key

Search binary tree if the input key exists in the tree

If exists, printf(“Exist number”)

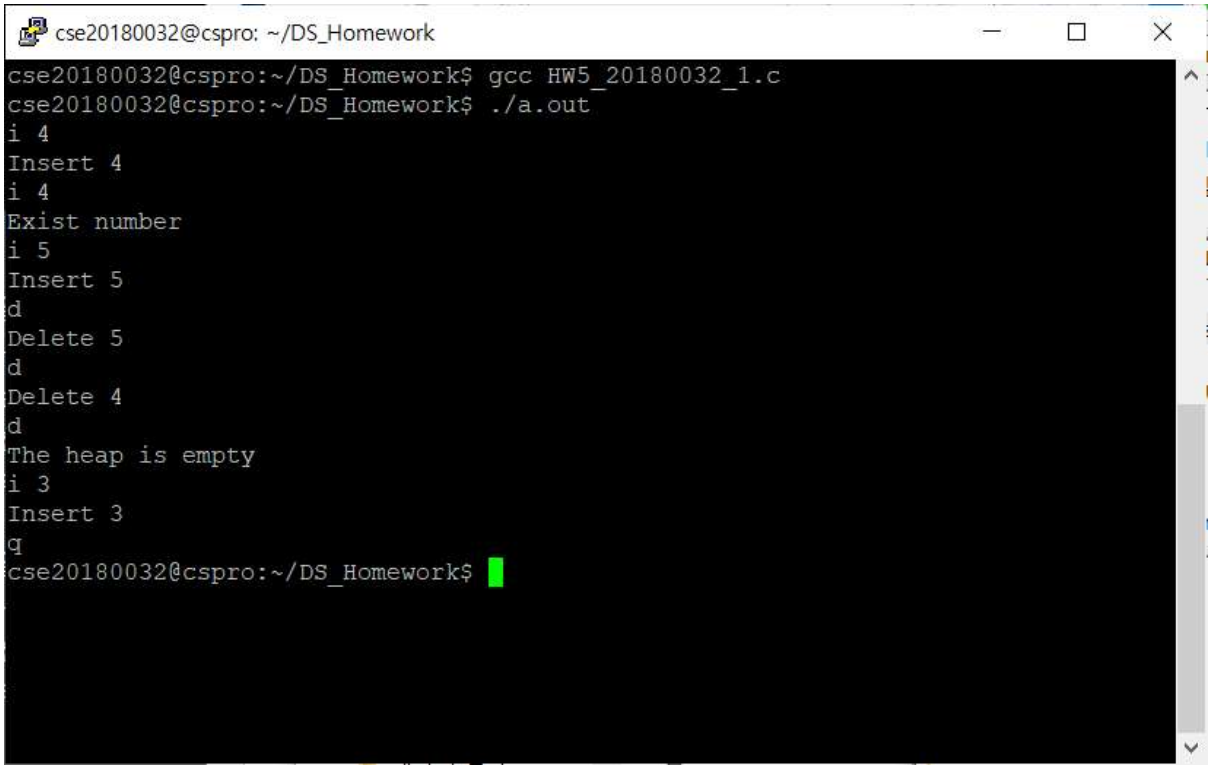
Else call Insert(key) and printf(“Insert %d”, key)

Case ‘d’ call Delete( )

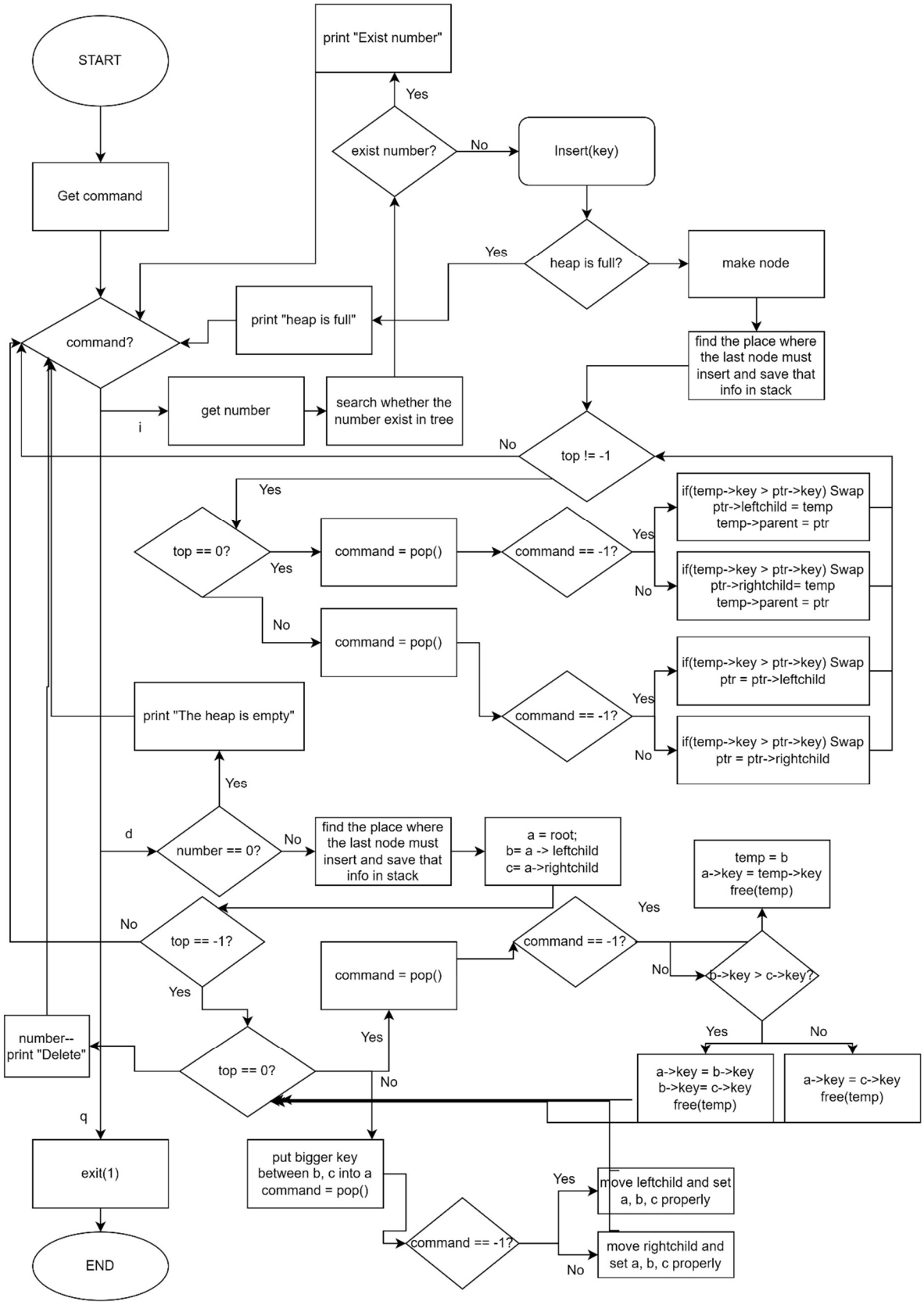
Case ‘q’ exit

}

<Test Examples>



<Flowchart>


START
Get command
command?
print "Exist number"
Yes
exist number?
No
Insert(key)
heap is full?
Yes
make node
print "heap is full"
find the place where the last node must insert and save that info in stack
get number
search whether the number exist in tree
i
top != -1
No
Yes
top == 0?
Yes
command = pop()
command == -1?
Yes
if(temp->key > ptr->key) Swap
ptr->leftchild = temp
temp->parent = ptr
No
if(temp->key > ptr->key) Swap
ptr->rightchild= temp
temp->parent = ptr
No
command = pop()
command == -1?
Yes
if(temp->key > ptr->key) Swap
ptr = ptr->leftchild
No
if(temp->key > ptr->key) Swap
ptr = ptr->rightchild
print "The heap is empty"
Yes
d
number == 0?
No
find the place where the last node must insert and save that info in stack
a = root;
b= a -> leftchild
c= a->rightchild
temp = b
a->key = temp->key
free(temp)
Yes
No
top == -1?
command = pop()
command == -1?
No
b->key > c->key?
Yes
Yes
No
number--
print "Delete"
top == 0?
Yes
No
a->key = b->key
b->key= c->key
free(temp)
a->key = c->key
free(temp)
q
exit(1)
END
put bigger key between b, c into a
command = pop()
command == -1?
Yes
move leftchild and set a, b, c properly
No
move rightchild and set a, b, c properly

2. Binary Search Tree(preorder, inorder, postorder)

<Pseudo Code>

Variable: int input[MAX_SIZE](save input keys), int stack[MAX_SIZE](using in reading preorder inputs), int top, tree_pointer root(root of binary search tree)

Function:

1. void make_tree_preorder(int num)

```
make first node 'temp'
push(input[0]);
root = temp; ptr= root
for (i=1; i< num; i++) {
        if( stack[top] > input[i] )
                make temp node and put input[i] to temp->key
                push(input[i])
                ptr->leftchild = temp; temp->parent = ptr;
                ptr = ptr->leftchild //move for next step
        else if(stack[top] < input[i] )
                while(stack[top] < input[i])
                        pop()
                        if(top != -1) ptr = ptr->parent;
                        if(top == -1) break;
                make temp node and put input[i] to temp->key
```

```
            push(input[i])
            ptr->rightchild = temp; temp->parent = ptr;
            ptr = ptr->rightchild //move for next step
```

2. void inorder(tree_pointer ptr)

```
If(ptr)
    inorder(ptr->leftchild)
    printf("%d ", ptr->key)
    inorder(ptr->rightchild)
```

3. void postorder(tree_pointer ptr)

```
If(ptr)
    postorder(ptr->leftchild)
    postorder(ptr->rightchild)
    printf("%d ", ptr->key)
```

4. main function

Get the number of tree nodes

Get key value of inputs and save it into array 'input[]'

Compare each value whether same keys exist

If same keys exist, printf("cannot construct BST") and end

Else

Make_tree_preorder(num);

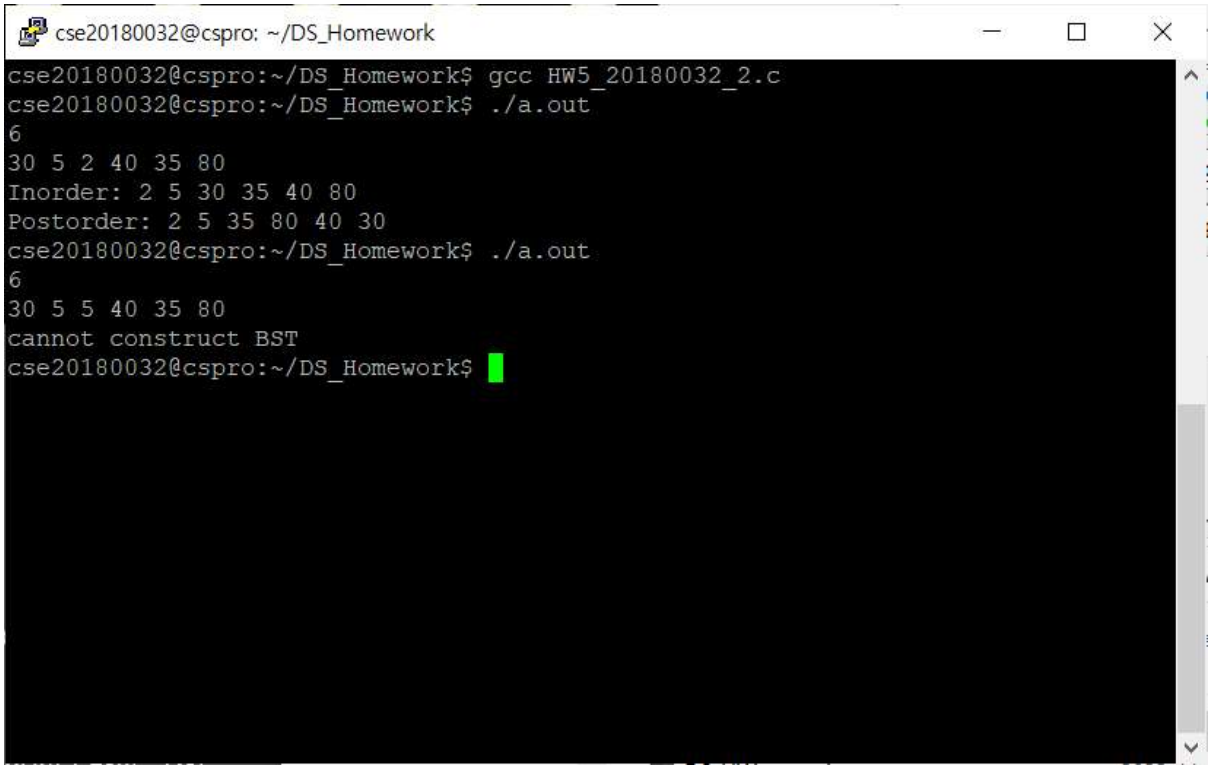
Printf("inorder: ");

inorder(root);

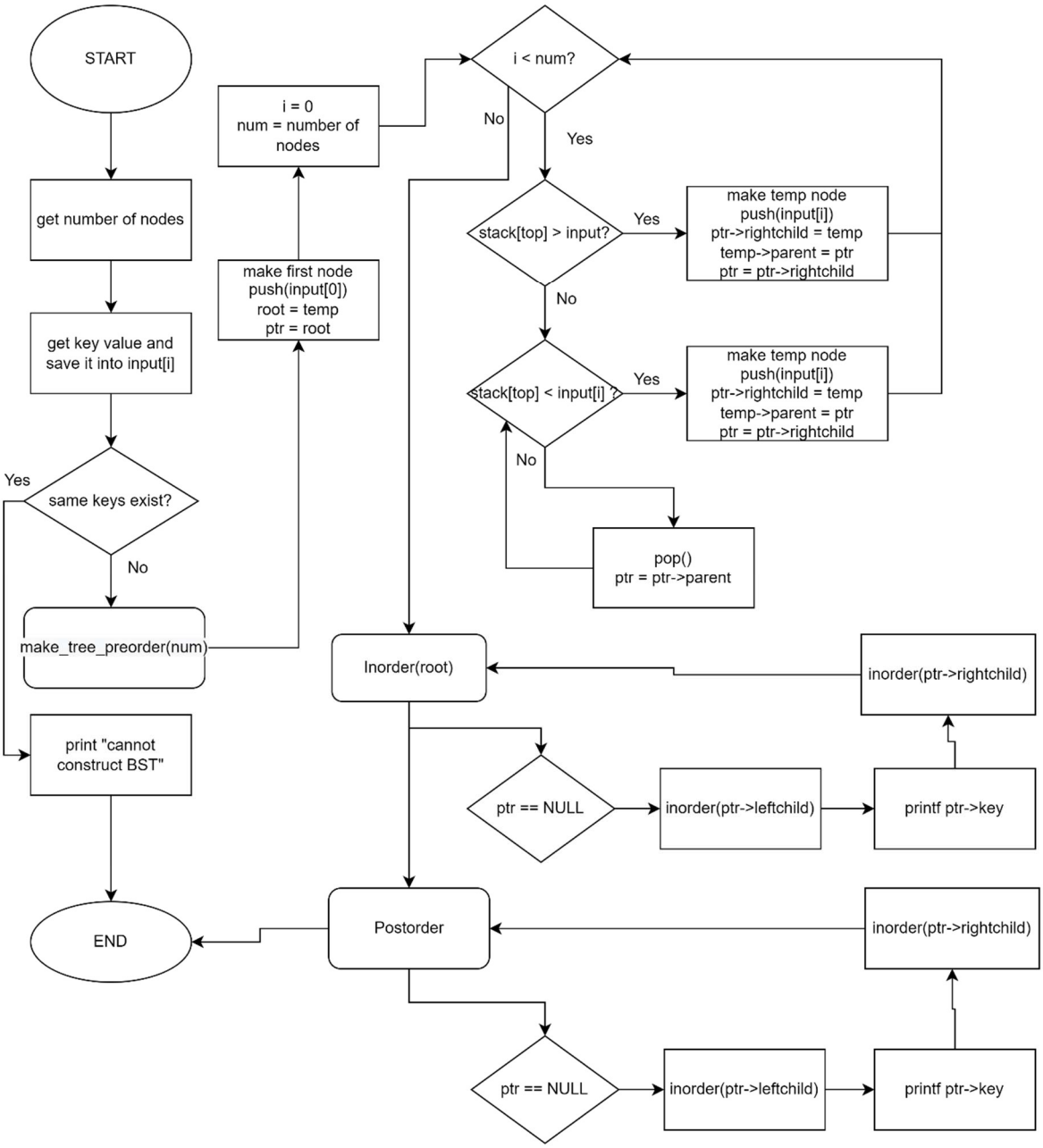
Printf("Postorder: ");

postorder(root):

<Test Examples>



<Flowchart>


START
get number of nodes
get key value and save it into input[i]
same keys exist?
Yes
No
make_tree_preorder(num)
print "cannot construct BST"
END
i = 0
num = number of nodes
make first node
push(input[0])
root = temp
ptr = root
i < num?
No
Yes
stack[top] > input?
Yes
No
make temp node
push(input[i])
ptr->rightchild = temp
temp->parent = ptr
ptr = ptr->rightchild
stack[top] < input[i] ?
Yes
No
make temp node
push(input[i])
ptr->rightchild = temp
temp->parent = ptr
ptr = ptr->rightchild
pop()
ptr = ptr->parent
Inorder(root)
ptr == NULL
inorder(ptr->leftchild)
printf ptr->key
inorder(ptr->rightchild)
Postorder
ptr == NULL
inorder(ptr->leftchild)
printf ptr->key
inorder(ptr->rightchild)

3. Max Priority Queue

<Pseudo Code>

Variable: root(root of binary search tree), key(number to insert)

Function: search, insert, pop, top, main

```
1. node_pointer search (int key)

If root == NULL, return NULL

for(ptr= root ; ; )

        if (ptr->key == key) return ptr;

        else

                if( leftchild exists) ptr= ptr->leftchild;

                else if( rightchild exists) ptr= ptr->rightchild;

                else return ptr // in this case the node is leaf


2. void insert (int key)

ptr = search(key)

if( key is found ) printf “Exist number” and return

temp = new node, temp->key = key, temp->leftchild=temp->rightchild=NULL;

if (root)

        if (temp->key < ptr->key)

                ptr->leftchild = temp, temp->parent=ptr, printf “Push ‘key’”
```

```
        else
                ptr->rightchild=temp, temp->parent=ptr, printf “Push ‘key’”
else
        temp->parent=NULL
        root = temp, printf “Push ‘key’”
```

3. void pop ( )

```
if( root == NULL) printf “The queue is empty”
int max = root->key
find greatest value in tree
stop at the node whose key is the greatest in tree(ptr)
if(ptr == root)
        if(leftchild exists) root = root->leftchild, printf “Pop ‘key’”, free(ptr)
        else if(rightchild exists) root = root->rightchild, printf “Pop ‘key’”, free(ptr)
        else(only root exists) printf “Pop ‘key’”, free(ptr), root = NULL;
else //greatest value is not root
        if( the greatest node is leaf )
                printf “Pop ‘key’”, ptr->parent->leftchild=NULL;
                ptr->parent->rightchild =NULL; free(ptr);
        else if (rightchild exists)
```

```
            trail = ptr->parent
            if(ptr->rightchild->key > trail-> key)
                trail->rightchild = ptr->rightchild, trail->leftchild =NULL;
                ptr->rightchild->parent = trail; printf “Pop”, free(ptr)
            else
                trail->leftchild = ptr->rightchild, trail->rightchild =NULL;
                ptr->rightchild->parent = trail; printf “Pop”, free(ptr)
        else if ( leftchild exists )
            trail = ptr->parent
            if(ptr->leftchild->key > trail->key)
                trail->rightchild = ptr->leftchild, trail->leftchild =NULL;
                ptr->leftchild->parent = trail; printf “Pop”, free(ptr)
            else
                trail->leftchild = ptr->leftchild, trail->rightchild =NULL;
                ptr->leftchild->parent = trail; printf “Pop”, free(ptr)
```

4. void top ( )

```
if( root == NULL) printf “The queue is empty”
int max = root->key
for(ptr= root ; ; )
```

if(ptr->key > max) max = ptr->key

if(leftchild exists) ptr= ptr->leftchild;

else if(rightchild exists) ptr= ptr->rightchild;

else break; //in this case the node is leaf

printf “The top is ’max’”

5. main function

for(; ;)

get command

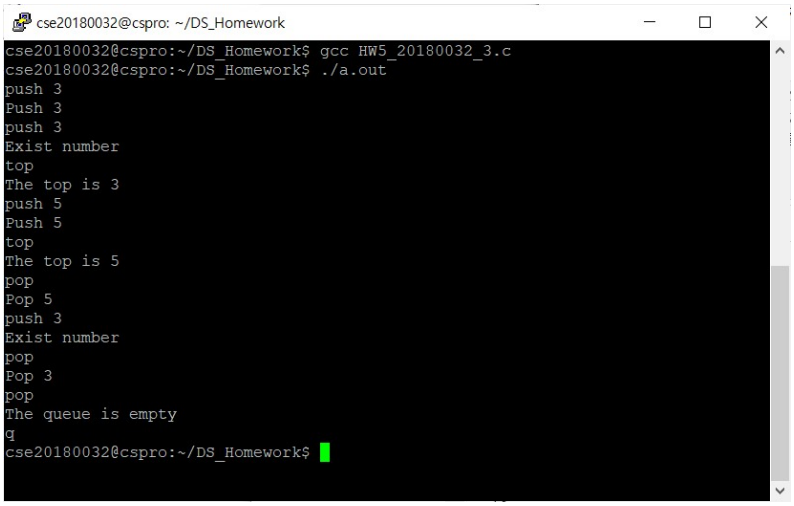
if (command is ‘push’) get ‘key’ and call insert(key)

else if (command is ‘top’) top( )

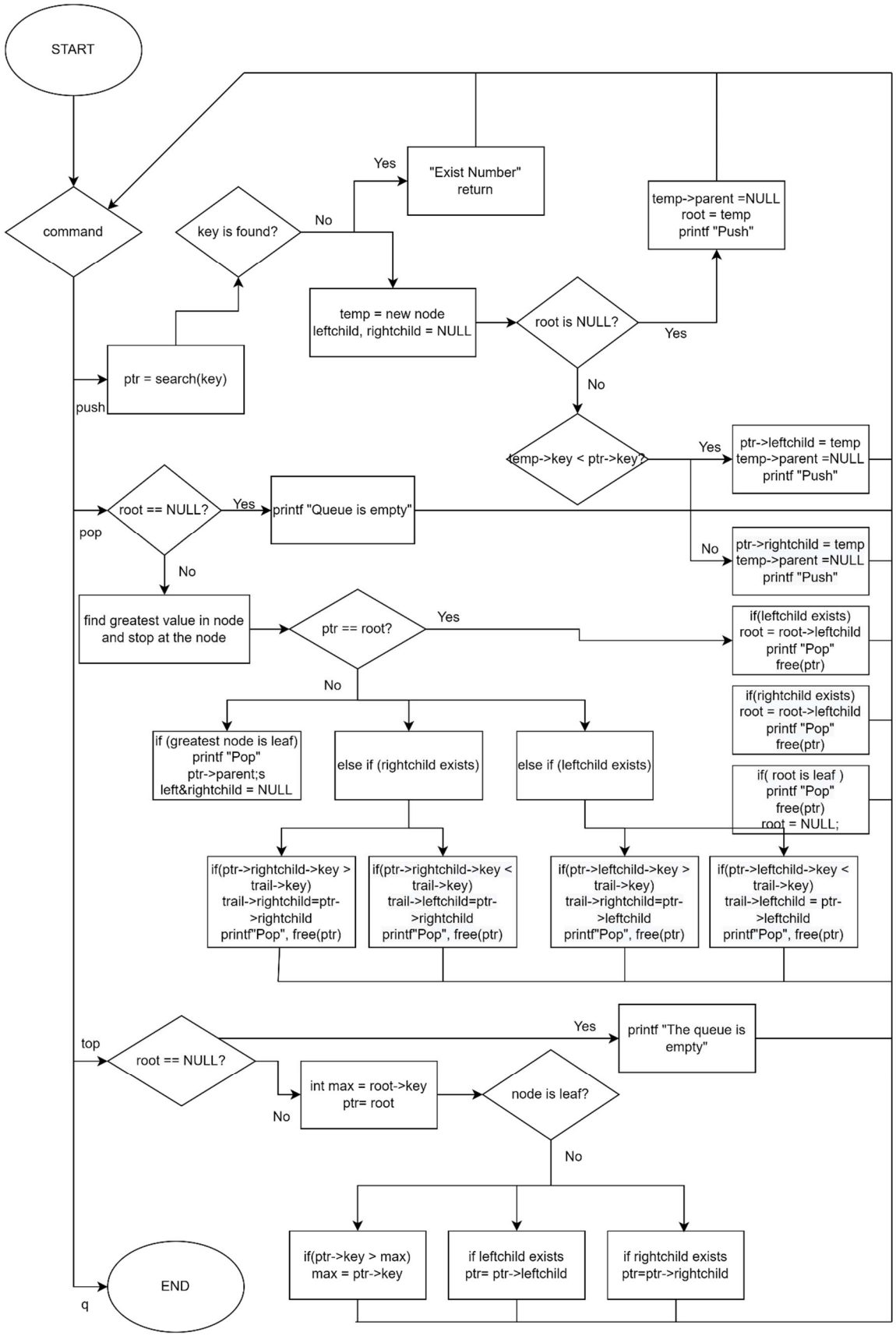
else if (command is ‘pop’) pop( )

else break;

<Test Examples>



<Flowchart>


START
command
key is found?
Yes
"Exist Number"
return
No
temp->parent =NULL
root = temp
printf "Push"
temp = new node
leftchild, rightchild = NULL
root is NULL?
Yes
No
ptr = search(key)
push
temp->key < ptr->key?
Yes
ptr->leftchild = temp
temp->parent =NULL
printf "Push"
root == NULL?
Yes
printf "Queue is empty"
pop
No
ptr->rightchild = temp
temp->parent =NULL
printf "Push"
No
find greatest value in node
and stop at the node
ptr == root?
Yes
if(leftchild exists)
root = root->leftchild
printf "Pop"
free(ptr)
No
if(rightchild exists)
root = root->leftchild
printf "Pop"
free(ptr)
if (greatest node is leaf)
printf "Pop"
ptr->parent;s
left&rightchild = NULL
else if (rightchild exists)
else if (leftchild exists)
if(root is leaf)
printf "Pop"
free(ptr)
root = NULL;
if(ptr->rightchild->key >
trail->key)
trail->rightchild=ptr-
>rightchild
printf"Pop", free(ptr)
if(ptr->rightchild->key <
trail->key)
trail->leftchild=ptr-
>rightchild
printf"Pop", free(ptr)
if(ptr->leftchild->key >
trail->key)
trail->rightchild=ptr-
>leftchild
printf"Pop", free(ptr)
if(ptr->leftchild->key <
trail->key)
trail->leftchild = ptr-
>leftchild
printf"Pop", free(ptr)
top
root == NULL?
Yes
printf "The queue is
empty"
No
int max = root->key
ptr= root
node is leaf?
No
if(ptr->key > max)
max = ptr->key
if leftchild exists
ptr= ptr->leftchild
if rightchild exists
ptr=ptr->rightchild
END
q