LAB6

Data Structure

Lab6

- The deadline for lab6 submission is 12th April at 11:59 pm.

- Folder name : lab6
- Code name: p6.c
- Each code will be tested by 5 different input files.
- 20 score for each input, if you don’t get the answer, you get 0 score.

Evaluation criteria

Category	Evaluation	
p6	100	
Total	100	

- *Use GCC **11** version.*
- *No score will be given if the gcc version is different.*

Lab6

- Design binary tree ADT and implement using array.
- Use binary tree ADT to perform pre-, in-, and post-order traversal using recursive functions.

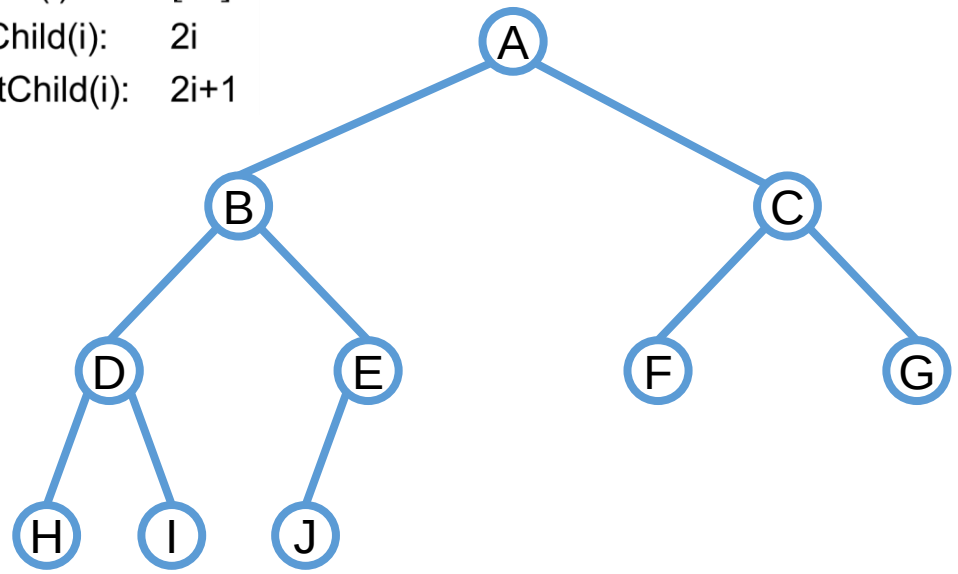
Lab6 – Tree

- Structure

```
struct TreeStruct{
        int size;
        int numOfNode;
        int* element;
};
typedef struct TreeStruct * Tree;
```

- Complete binary tree

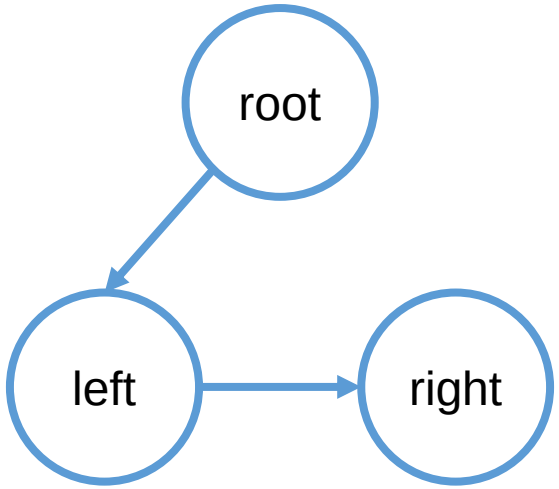
i is index
parent(i): [i/2]
leftChild(i): 2i
rightChild(i): 2i+1



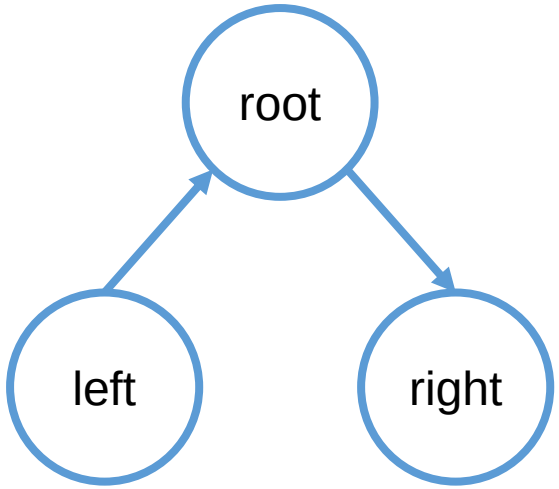
	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]
	A	B	C	D	E	F	G	H	I	J

Lab6 – Tree

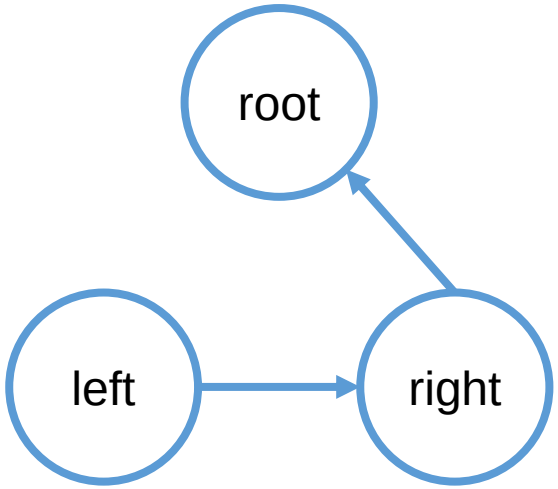
- Preorder



- Inorder



- Postorder



Lab6 – Tree

- Structure

```
struct TreeStruct{
        int size;
        int numOfNode;
        int* element;
};
typedef struct TreeStruct * Tree;
```

- Function

```
<Lab6>
Tree CreateTree(int size);
void Insert(Tree tree, int value);
void printTree(Tree tree);
void printPreorder(Tree tree, int index);
void printInorder(Tree tree, int index);
void printPostorder(Tree tree, int index);
void DeleteTree(Tree tree);
```

Lab6 – Tree

- **CreateTree** create a new complete binary tree with tree size.
- **Insert** insert a new node in tree. If tree is full, just print an error message.
- **PrintTree** print the tree by traversals.
- **PrintInorder** print the tree by inorder traversal.
- **PrintPreorder** print the tree by preorder traversal.
- **PrintPostorder** print the tree by postorder traversal.
- **DeleteTree** free all the memory allocated to tree.

Lab6 – Tree

- input file : input1.txt

```
10
1 2 3 4 5 6 7 8
```

- Result

```
Preorder: 1 2 4 8 5 3 6 7
Inorder: 8 4 2 5 1 6 3 7
Postorder: 8 4 5 2 6 7 3 1
```

- input file : input2.txt

```
5
1 2 3 4 5 6 7
```

- Result

```
Error! Tree is full.
Error! Tree is full.
Preorder: 1 2 4 5 3
Inorder: 4 2 5 1 3
Postorder: 4 5 2 3 1
```

Lab6 – Tree

```
#include<stdio.h>
#include<stdlib.h>

struct TreeStruct{
	int size;
	int numOfNode;
	int* element;
};

typedef struct TreeStruct* Tree;

Tree CreateTree(int size);
void Insert(Tree tree, int value);
void PrintTree(Tree tree);
void PrintPreorder(Tree tree, int index);
void PrintInorder(Tree tree, int index);
void PrintPostorder(Tree tree, int index);
void DeleteTree(Tree tree);
```

```
void main(int argc, char* argv[])
{
	FILE *fi;
	Tree tree;
	int treeSize;
	int tmpNum;

	fi = fopen(argv[1], "r");
	fscanf(fi, "%d", &treeSize);
	tree = CreateTree(treeSize);

	while(fscanf(fi, "%d", &tmpNum) == 1)
	{
		Insert(tree, tmpNum);
	}
	PrintTree(tree);
	DeleteTree(tree);
}
```

Lab6 – Tree

- program name : p6.c
- input : a list of numbers in a file.
- output : the corresponding result in the standard output.