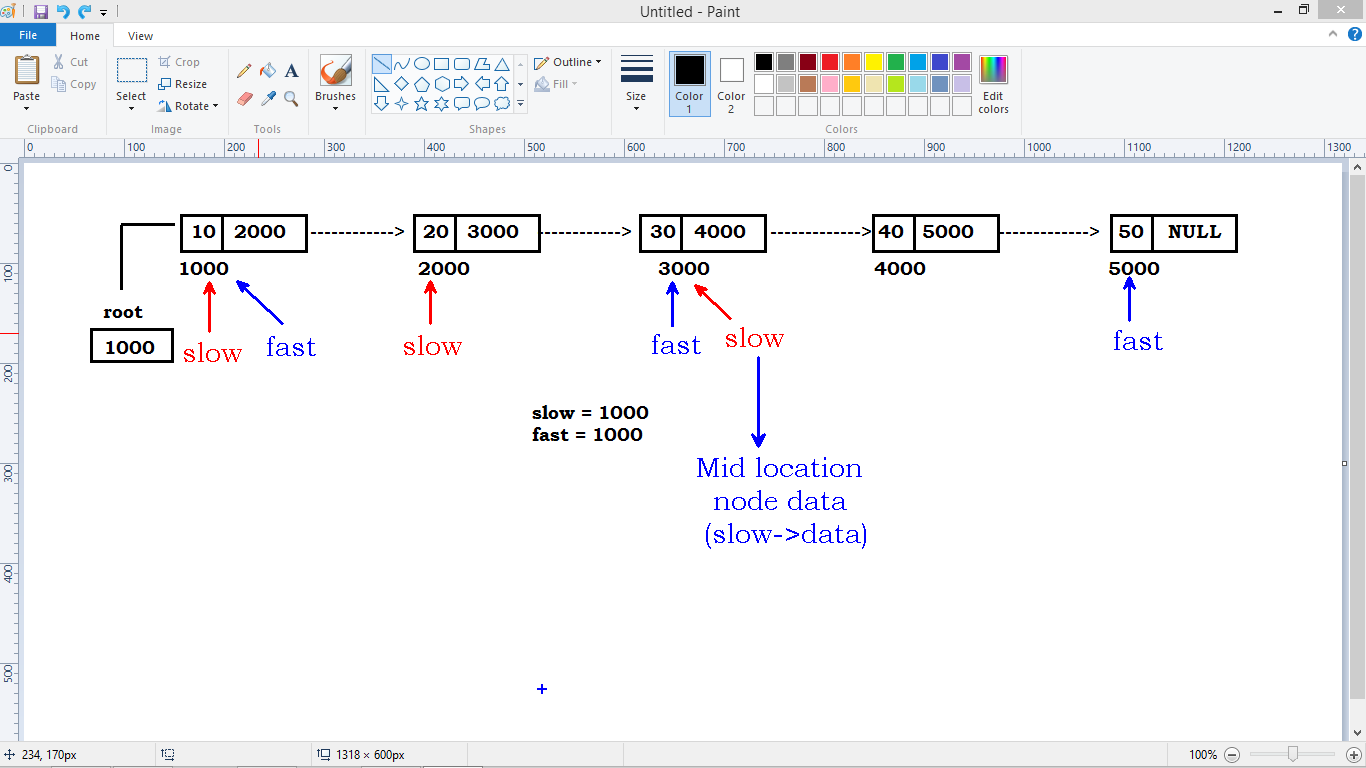
**Slow and Fast Pointer**

**Question: How to find Middle element in Linked List?**

**Answer: Using Fast pointer.**

****

* **Insert 10 nodes into list using recursion**
* **Node values 10,20,30,40,50,60,70,80,90,100 (for i=10 ; i<=100 ; i=i+10)**
* **Find the mid location element using fast pointer**

**#include<stdio.h>**

**#include<stdlib.h>**

**struct Node**

**{**

**int data;**

**struct Node\* link;**

**};**

**struct Node\* root=NULL;**

**int length();**

**struct Node\* create(int);**

**struct Node\* insert(struct Node\*,int);**

**int middle();**

**void display();**

**int main()**

**{**

**int i,x;**

**for(i=10 ; i<=100 ; i+=10)**

**{**

**root = insert(root, i);**

**}**

**x = middle();**

**printf("Middle value : %d\n",x);**

**return 0;**

**}**

**struct Node\* create(int ele)**

**{**

**struct Node\* temp;**

**temp=(struct Node\*)malloc(sizeof(struct Node));**

**temp->data=ele;**

**temp->link=NULL;**

**return temp;**

**}**

**struct Node\* insert(struct Node\* temp, int ele)**

**{**

**if(temp==NULL)**

**{**

**return create(ele);**

**}**

**else**

**{**

**temp->link = insert(temp->link,ele);**

**}**

**return temp;**

**}**

**int middle()**

**{**

**struct Node \*fast, \*slow;**

**slow=fast=root;**

**while(fast->link!=NULL && fast->link->link!=NULL)**

**{**

**slow=slow->link;**

**fast=fast->link->link;**

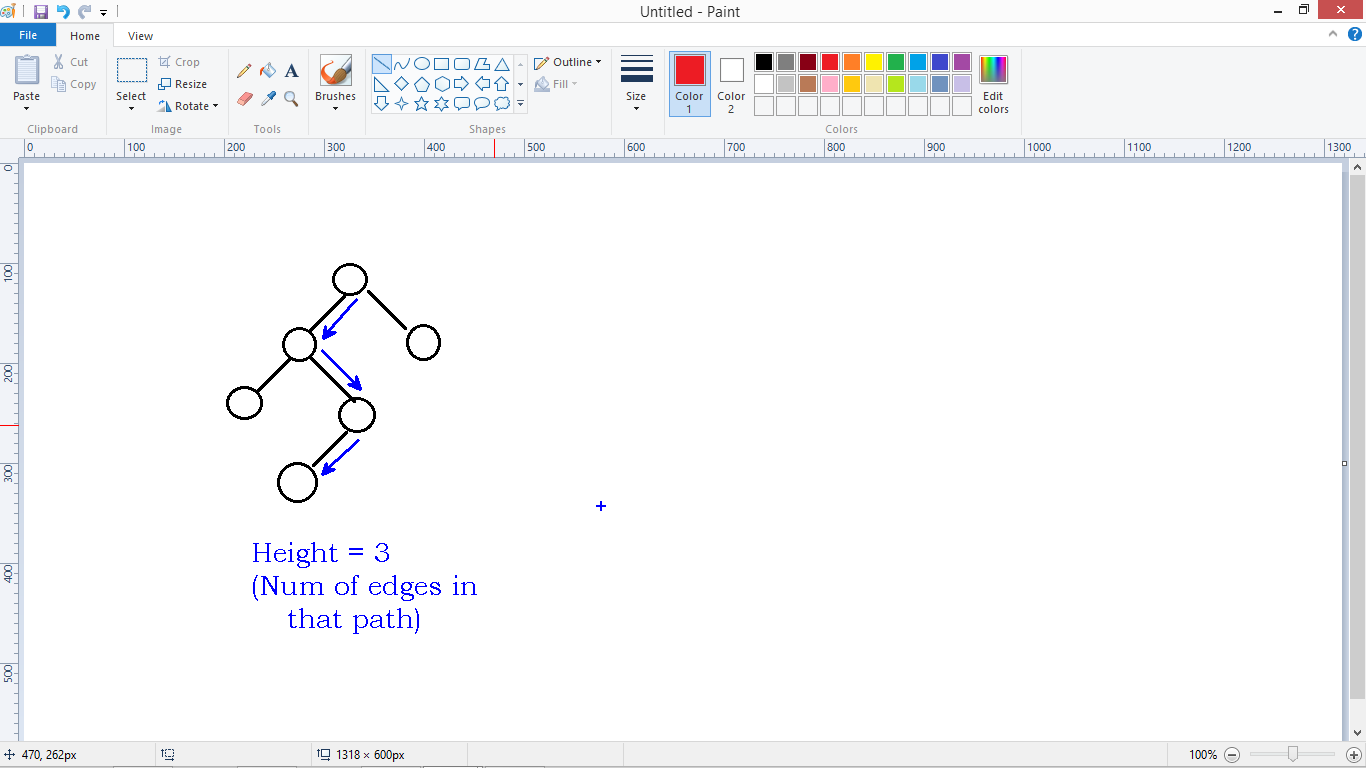
**}**

**return slow->data;**

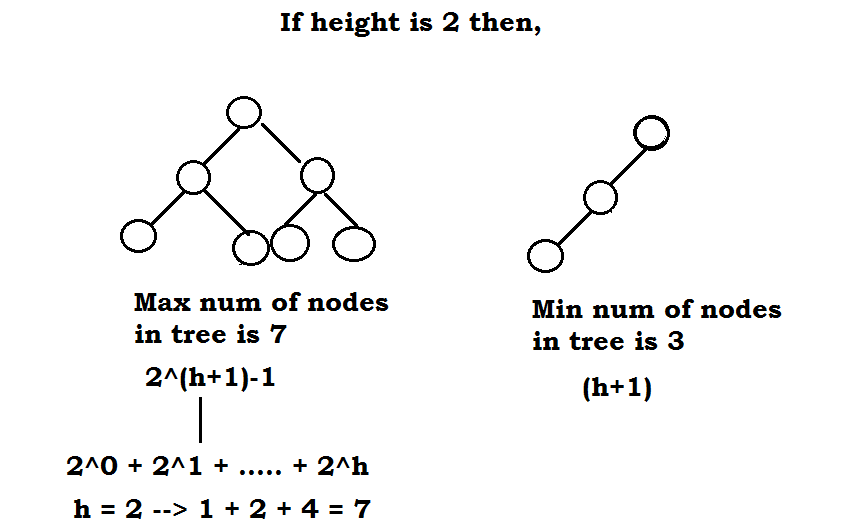
**}**

**Find the height of BST:**

* **The height of BST is the longest path from its root node any leaf node.**

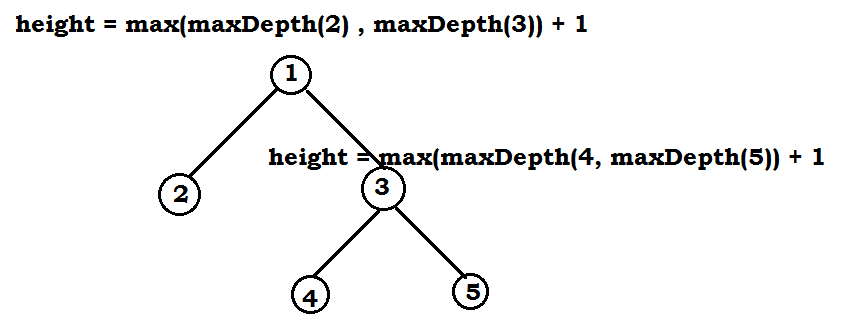
****

* **How to find the height with minimum nodes and maximum nodes:**
* **We can find the min and max number of nodes using height of the tree as follows**

****

* Find the maximum depth of left sub tree and right sub tree first.
* Find the Biggest value among left sub tree and right sub tree.
* Biggest value + 1 is nothing but the depth of the Root node

**Recursion call:**



**Code:**

#include<stdio.h>

#include<stdlib.h>

struct Node

{

int data;

struct Node \*left;

struct Node \*right;

};

struct Node\* root = NULL;

struct Node\* insert(struct Node\*, int);

struct Node\* new\_node(int);

void print(struct Node\*);

int height(struct Node\*);

int main()

{

int i, ht;

for(i=10 ; i<=50 ; i=i+10)

{

root = insert(root , i);

}

printf("Elements are : \n");

print(root);

printf("\n");

ht = height(root);

printf("Height of tree is : %d \n", ht);

return 0;

}

struct Node\* insert(struct Node\* node, int ele)

{

if(node == NULL)

{

return new\_node(ele);

}

if(ele < node->data)

{

node->left = insert(node->left , ele);

}

if(ele > node->data)

{

node->right = insert(node->right , ele);

}

return node;

}

struct Node\* new\_node(int ele)

{

struct Node\* node;

node = (struct Node\*)malloc(sizeof(struct Node));

node->data = ele;

node->left = NULL;

node->right = NULL;

return node;

}

void print(struct Node\* node)

{

if(node->left)

{

print(node->left);

}

printf("%d \t", node->data);

if(node->right)

{

print(node->right);

}

}

int height(struct Node\* node)

{

int lheight, rheight;

if(!node)

{

return 0;

}

else

{

lheight = height(node->left);

rheight = height(node->right);

if(lheight >= rheight)

return lheight+1;

else

return rheight+1;

}

}

**Explanation:**

