An algorithm is a step by step procedure describing the solution of a given problem. The following algorithm describes inserting a

```
Algorithm to Insert a Node Into a Tree

Step 1:Begin Algorithm

Step 2:Define head node and number

Step 3:Check if tree exists
         if tree does not exists
           allocate memory and assign begining address to head
           assign number to head -> a
           assign NULL to head -> left and head -> right
         else
           while temp != NULL, do the following
             if number > temp -> a
               prev = temp
               temp = temp -> right
             else
               prev = temp
               temp = temp -> left

Step 4:Allocate new memory & assign beginning address to temp
         temp = (struct node *)malloc(sizeof(struct node));

Step 5:Assign number to temp -> a

Step 6:Check if number >= prev -> a
         if (number >= prev->a)
           prev -> right = temp
         else
           prev->left = temp;

Step 7:Return to main program
```

```
// C function for inserting a node to a tree
//
void insert(struct node **head, int num)
{
    struct node *temp = *head, *prev = *head;

    if (*head == NULL)
    {
        *head = (struct node *)malloc(sizeof(struct node));
        (*head)->a = num;
        (*head)->left = (*head)->right = NULL;
    }
    else
    {
        while (temp != NULL)
        {
            if (num > temp->a)
            {
                prev = temp;
                temp = temp->right;
            }
            else
            {
                prev = temp;
                temp = temp->left;
            }
        }
        temp = (struct node *)malloc(sizeof(struct node));
        temp->a = num;
        if (num >= prev->a)
        {
            prev->right = temp;
        }
        else
        {
            prev->left = temp;
        }
    }
}
```

An algorithm is a step by step procedure describing the solution of a given problem. The following algorithm describes inserting a

```
Algorithm to Insert a Node Into a Tree

Step 8:Begin Algorithm

Step 9:Define head node and number

Step 10:  Check if tree exists
          if tree does not exists
            allocate memory and assign begining address to head
            assign number to head -> a
            assign NULL to head -> left and head -> right
          else
            while temp != NULL, do the following
              if number > temp -> a
                prev = temp
                temp = temp -> right
              else
                prev = temp
                temp = temp -> left

Step 11:  Allocate new memory & assign beginning address to
        temp
            temp = (struct node *)malloc(sizeof(struct node));

Step 12:  Assign number to temp -> a

Step 13:  Check if number >= prev -> a
          if (number >= prev->a)
            prev -> right = temp
          else
            prev->left = temp;

Step 14:  Return to main program
```

ALGORITHM TO INSERT A NODE TO THE TREE