```c
// 1. Using linked list representation, create a
// binary tree for the expression:
// 5 * ( (3 * 4) + (6 / 2 ) - (8 + 2) )
#include <stdio.h>
#include <stdlib.h>

typedef struct node
{
   char data;
   struct node *left;
   struct node *right;
} Node;

Node *createNode(char data)
{
   Node *newNode = malloc(sizeof(Node));
   newNode->data = data;
   newNode->left = newNode->right = NULL;
   return newNode;
}

void inorder(Node *root)
{
   if (!root)
       return;
   inorder(root->left);
   printf(" %c ", root->data);
   inorder(root->right);
}
int main()
{
   Node *five = createNode('5');
```

```
    Node *two = createNode('2');
    Node *six = createNode('6');
    Node *eight = createNode('8');
    Node *three = createNode('3');
    Node *four = createNode('4');
    Node *two2 = createNode('2');

    Node *mul1 = createNode('*');
    mul1->left = three;
    mul1->right = four;

    Node *plus1 = createNode('+');
    plus1->left = eight;
    plus1->right = two;

    Node *div = createNode('/');
    div->left = six;
    div->right = two2;

    Node *plus2 = createNode('+');
    plus2->left = mul1;
    plus2->right = div;

    Node *sub = createNode('-');
    sub->left = plus2;
    sub->right = plus1;

    Node *root = createNode('*');
    root->left = five;
    root->right = sub;

    printf("Result: ");
```

```c
   inorder(root);
   return 0;
}

// 2. For the tree created in Program 1, that is,
// Write the program for the function(s) to traverse the
//  tree in preorder, postorder and
// inorder traversal.
#include <stdio.h>
#include <stdlib.h>

typedef struct node
{
   char data;
   struct node *left;
   struct node *right;
} Node;

Node *createNode(char data)
{
   Node *newNode = malloc(sizeof(Node));
   newNode->data = data;
   newNode->left = newNode->right = NULL;
   return newNode;
}

void preorder(Node *root)
{
   if (!root)
       return;

   printf(" %c ", root->data);
```

```c
    preorder(root->left);
    preorder(root->right);
}

void inorder(Node *root)
{
    if (!root)
        return;

    if (root->left || root->right)
    {
        printf("(");
    }
    inorder(root->left);
    printf(" %c ", root->data);
    inorder(root->right);
    if (root->left || root->right)
    {
        printf(")");
    }
}

void postorder(Node *root)
{
    if (!root)
        return;

    postorder(root->left);
    postorder(root->right);
    printf(" %c ", root->data);
}
```

```c
int main()
{
   Node *five = createNode('5');
   Node *two = createNode('2');
   Node *six = createNode('6');
   Node *eight = createNode('8');
   Node *three = createNode('3');
   Node *four = createNode('4');
   Node *two2 = createNode('2');

   Node *mul1 = createNode('*');
   mul1->left = three;
   mul1->right = four;

   Node *plus1 = createNode('+');
   plus1->left = eight;
   plus1->right = two;

   Node *div = createNode('/');
   div->left = six;
   div->right = two2;

   Node *plus2 = createNode('+');
   plus2->left = mul1;
   plus2->right = div;

   Node *sub = createNode('-');
   sub->left = plus2;
   sub->right = plus1;

   Node *root = createNode('*');
   root->left = five;
   root->right = sub;

   printf("\n Preorder Expression: ");
   preorder(root);
   printf("\n Inorder Expression: ");
   inorder(root);
   printf("\n Postorder Expression: ");
   postorder(root);

   return 0;
}

// 3. For the tree created in program 1, write a program
// for counting the number of leaf nodes.
#include <stdio.h>
#include <stdlib.h>

typedef struct node
{
   char data;
   struct node *left;
   struct node *right;
} Node;

Node *createNode(char data)
{
   Node *newNode = malloc(sizeof(Node));
   newNode->data = data;
   newNode->left = newNode->right = NULL;
   return newNode;
}
```

```
int countNode(Node *n)
{
   if (!n)
       return 0;

   return 1 + (countNode(n->left) + countNode(n->right));
}

int main()
{
   Node *five = createNode('5');
   Node *two = createNode('2');
   Node *six = createNode('6');
   Node *eight = createNode('8');
   Node *three = createNode('3');
   Node *four = createNode('4');
   Node *two2 = createNode('2');

   Node *mul1 = createNode('*');
   mul1->left = three;
   mul1->right = four;

   Node *plus1 = createNode('+');
   plus1->left = eight;
   plus1->right = two;

   Node *div = createNode('/');
   div->left = six;
   div->right = two2;

   Node *plus2 = createNode('+');
   plus2->left = mul1;
```

```c
   plus2->right = div;

   Node *sub = createNode('-');
   sub->left = plus2;
   sub->right = plus1;

   Node *root = createNode('*');
   root->left = five;
   root->right = sub;

   printf("Total Nodes: %d", countNode(root));

   return 0;
}

// 4. Write a program to evaluate the expression
// given in Program 1, using its tree.
#include <stdio.h>
#include <stdlib.h>

typedef struct node
{
   char data;
   struct node *left;
   struct node *right;
} Node;

Node *createNode(char data)
{
   Node *newNode = malloc(sizeof(Node));
   newNode->data = data;
   newNode->left = newNode->right = NULL;
```

```
    return newNode;
}

int isOperator(char c)
{

    return (c == '+' || c == '-' || c == '*' || c == '/');
}

int calculate(Node *root)
{
    if (!root)
        return 0;
    if (!isOperator(root->data))
        return root->data - '0';
    int leftval = calculate(root->left);
    int rightval = calculate(root->right);
    switch (root->data)
    {
    case '+':
        return leftval + rightval;

    case '-':
        return leftval - rightval;

    case '*':
        return leftval * rightval;

    case '/':
        return leftval / rightval;
    }
    return 0;
```

```
}

int main()
{
   Node *five = createNode('5');
   Node *two = createNode('2');
   Node *six = createNode('6');
   Node *eight = createNode('8');
   Node *three = createNode('3');
   Node *four = createNode('4');
   Node *two2 = createNode('2');

   Node *mul1 = createNode('*');
   mul1->left = three;
   mul1->right = four;

   Node *plus1 = createNode('+');
   plus1->left = eight;
   plus1->right = two;

   Node *div = createNode('/');
   div->left = six;
   div->right = two2;

   Node *plus2 = createNode('+');
   plus2->left = mul1;
   plus2->right = div;

   Node *sub = createNode('-');
   sub->left = plus2;
   sub->right = plus1;
```

```c
    Node *root = createNode('*');
    root->left = five;
    root->right = sub;

    printf("Result: %d\n", calculate(root));

    return 0;
}
```