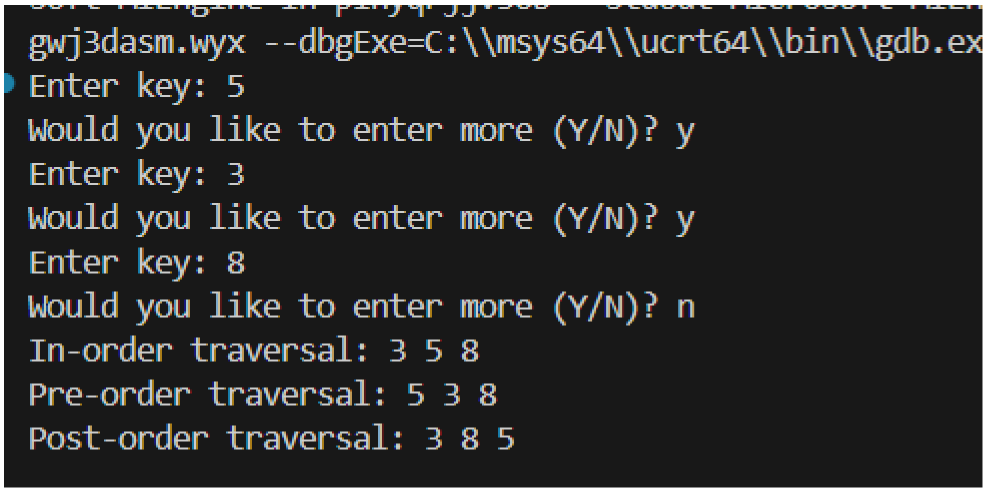
```c
#include <stdio.h>
#include <stdlib.h>

struct Node {
    int key;
    struct Node* left;
    struct Node* right;
};

struct Node* newNode(int key) {
    struct Node* temp = (struct Node*) malloc(sizeof(struct Node));
    temp->key = key;
    temp->left = temp->right = NULL;
    return temp;
}

struct Node* insert(struct Node* node, int key) {
    if (node == NULL)
        return newNode(key);

    if (key < node->key)
        node->left = insert(node->left, key);
    else if (key > node->key)
        node->right = insert(node->right, key);

    return node;
}

void inorder(struct Node* root) {
    if (root != NULL) {
        inorder(root->left);
        printf("%d ", root->key);
        inorder(root->right);
    }
}

void preorder(struct Node* root) {
    if (root != NULL) {
        printf("%d ", root->key);
        preorder(root->left);
        preorder(root->right);
    }
}

void postorder(struct Node* root) {
    if (root != NULL) {
        postorder(root->left);
        postorder(root->right);
        printf("%d ", root->key);
    }
}

int main() {
    struct Node* root = NULL;
    int key, c;

    while (1) {
        printf("Enter key: ");
        scanf("%d", &key);
        root = insert(root, key);
        printf("Would you like to enter more (Y/N)? ");
        scanf(" %c", &c);  // Adding space before %c to consume the newline
character
        if (c == 'N' || c == 'n')
            break;
    }

    printf("In-order traversal: ");
    inorder(root);
    printf("\n");

    printf("Pre-order traversal: ");
    preorder(root);
    printf("\n");

    printf("Post-order traversal: ");
    postorder(root);
    printf("\n");

    return 0;
}
```

Output:



```
gwj3dasm.wyx --dbgExe=C:\\msys64\\ucrt64\\bin\\gdb.ex
Enter key: 5
Would you like to enter more (Y/N)? y
Enter key: 3
Would you like to enter more (Y/N)? y
Enter key: 8
Would you like to enter more (Y/N)? n
In-order traversal: 3 5 8
Pre-order traversal: 5 3 8
Post-order traversal: 3 8 5
```

8. Binary Search Tree Implementation and Traversal

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

struct Node
 {
    int key;
    struct Node*left, *right;
 }

struct Node* newNode (int key)
 {
    struct Node* temp = (struct Node*) malloc (sizeof(Node));
    temp -> key = key;
    temp -> left = temp -> right = NULL;
    return temp;
 }

struct Node* insert (struct Node* node, int key)
 {
    if (node == NULL)
        return newNode (key);

    if ( key < node -> key)
        node -> left = insert (node -> left, key);
    else if ( key > node -> key)
        node -> right = insert (node -> right , key);

    return node;
 }
```

```
void inorder (struct Node* root)
{
    if (root ! = NULL)
     {
        inorder (root -> left),
        printf ("%d", root -> key);
        inorder (root -> right);
     }
}

void preorder (struct Node* root)
 {
    if (root ! = NULL)
     {
        printf("%d", root -> key);
        preorder (root -> left);
        preorder (root -> right);
     }
}

void postorder (struct Node* root)
{
    if (root*! = NULL)
     {
        postorder (root -> left);
        postorder (root -> right);
        printf ("%d", root -> key);
     }
}
```

PAGE
DATE / /

```
int main()
{
    struct Node* root = NULL;
    int key~~s~~, ~~e~~ char c;
    while (1)
    {
        printf("Enter key : ");
        scanf("%d", &key);
        root = insert (root, keys[i]);
        printf ("Would you like to enter more [Y/N] ? ");
        scanf(" %c", &c);
        if (c == 'N' || c == 'n');
            break;
    }
    printf ("In-order traversal:");
    inorder (root);
    printf("\n");
    printf (" Pre-order traversal:");
    preorder (root);
    printf("\n");
    printf ("Post-order traversal :");
    postorder (root);
    printf("\n");
    return 0;
}
```

Output:

```
Enter key : 10                  | In-order traversal: 10 20 30 50
Enter more [Y/n] : Y            | Pre-order traversal: 10 50 20 30
Enter key : 50                  | Post order traversal: 30 20 50 1
Enter more [Y/n] ; Y            |
Enter key 20                    |
Enter more [Y/n] : Y            |
Enter key 30                    |
Enter more [Y/n] : n            |
```