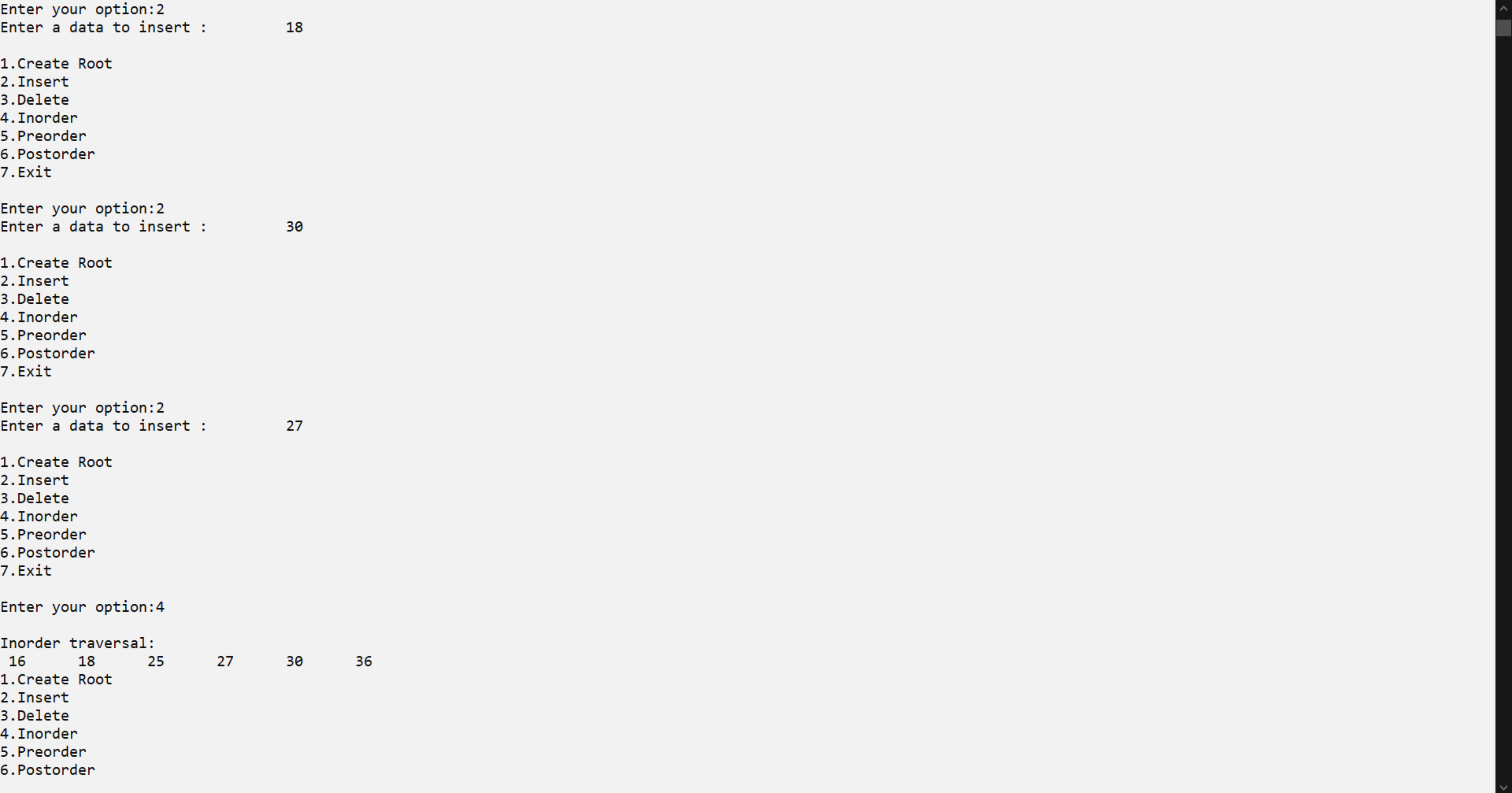
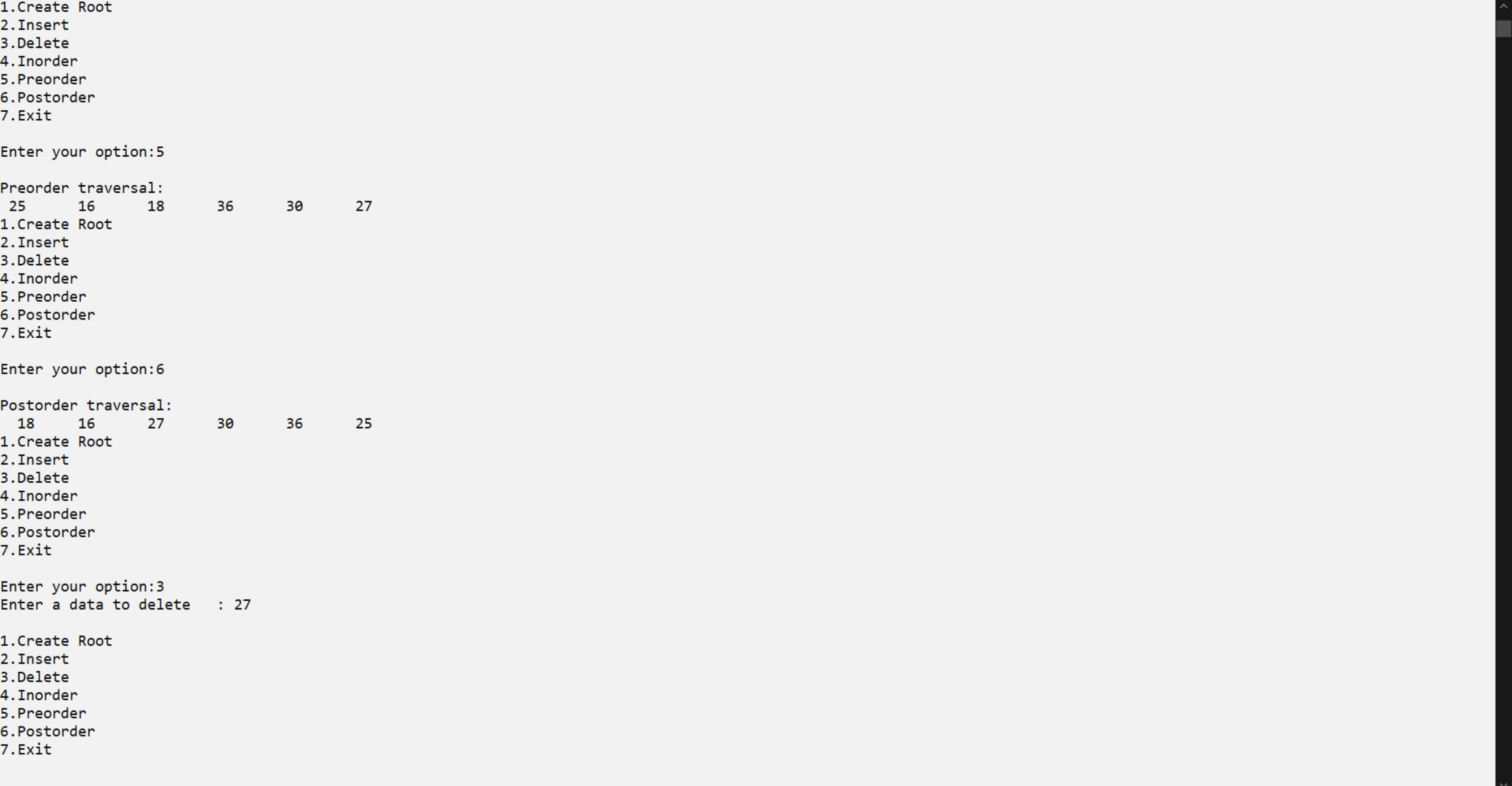
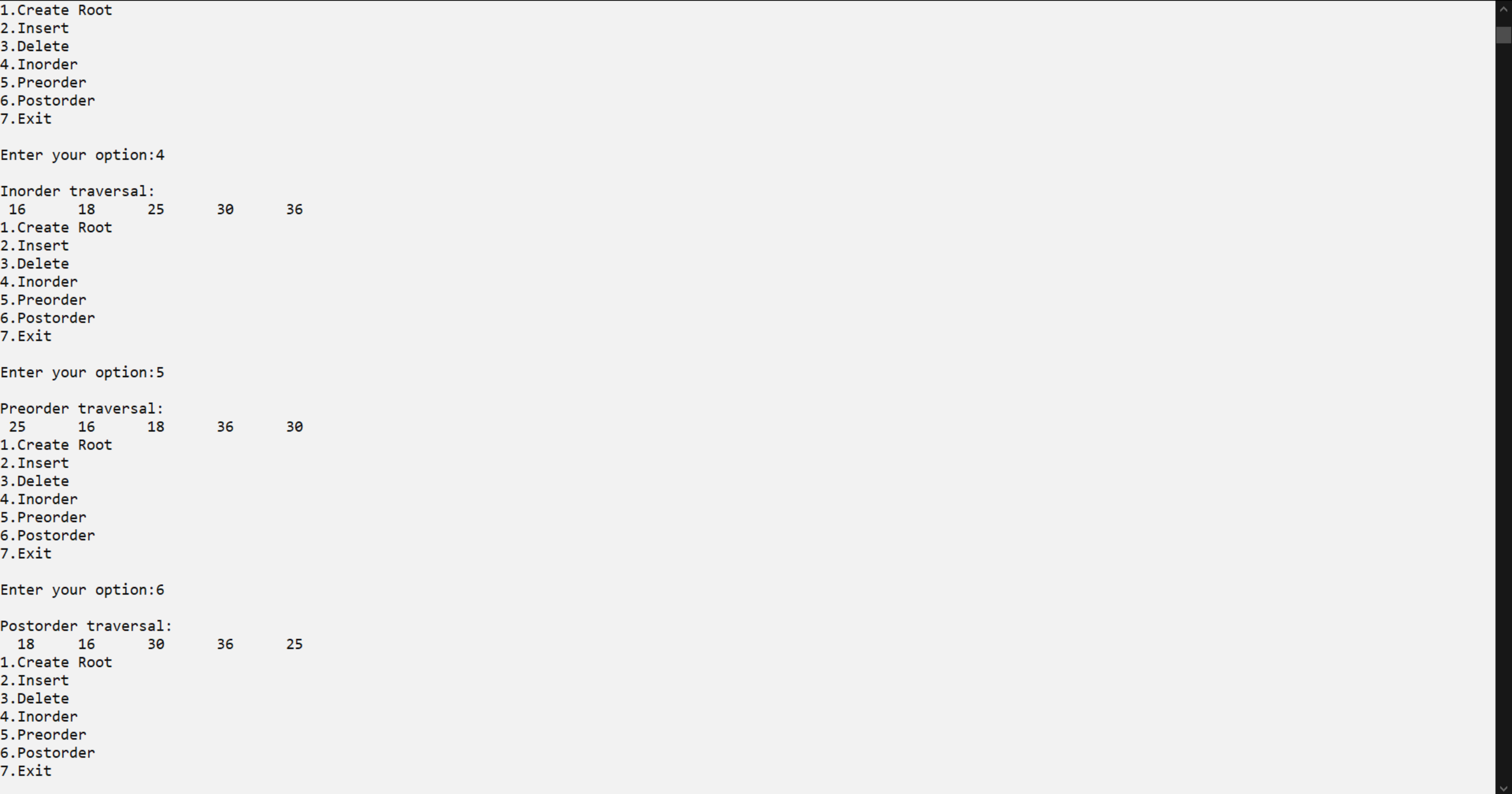
```
1.Create Root
2.Insert
3.Delete
4.Inorder
5.Preorder
6.Postorder
7.Exit

Enter your option:1
Enter the data :        25

1.Create Root
2.Insert
3.Delete
4.Inorder
5.Preorder
6.Postorder
7.Exit

Enter your option:2
Enter a data to insert :         16

1.Create Root
2.Insert
3.Delete
4.Inorder
5.Preorder
6.Postorder
7.Exit

Enter your option:2
Enter a data to insert :         36

1.Create Root
2.Insert
3.Delete
4.Inorder
5.Preorder
6.Postorder
7.Exit

Enter your option:2
```



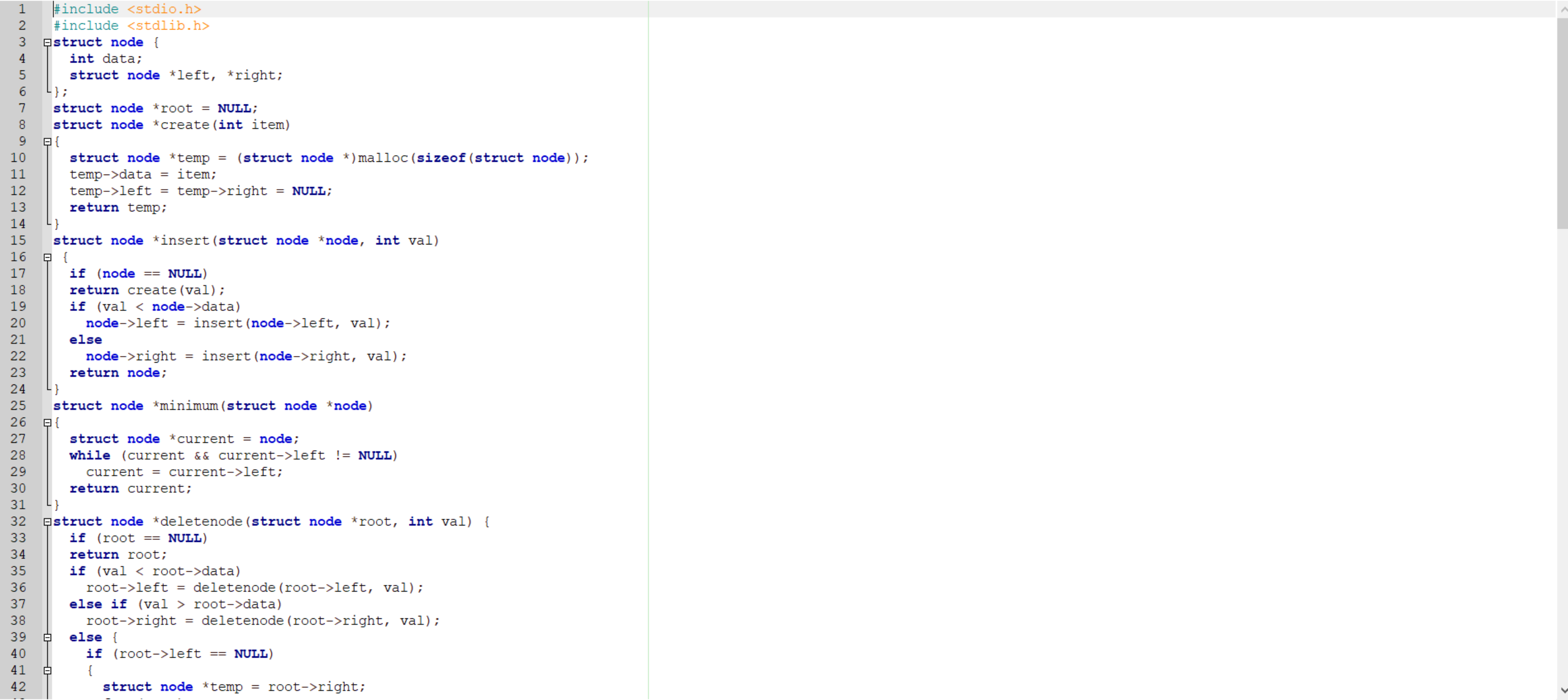
```
Enter your option:2
Enter a data to insert :         18

1.Create Root
2.Insert
3.Delete
4.Inorder
5.Preorder
6.Postorder
7.Exit

Enter your option:2
Enter a data to insert :         30

1.Create Root
2.Insert
3.Delete
4.Inorder
5.Preorder
6.Postorder
7.Exit

Enter your option:2
Enter a data to insert :         27

1.Create Root
2.Insert
3.Delete
4.Inorder
5.Preorder
6.Postorder
7.Exit

Enter your option:4

Inorder traversal:
 16      18      25      27      30      36
1.Create Root
2.Insert
3.Delete
4.Inorder
5.Preorder
6.Postorder
```



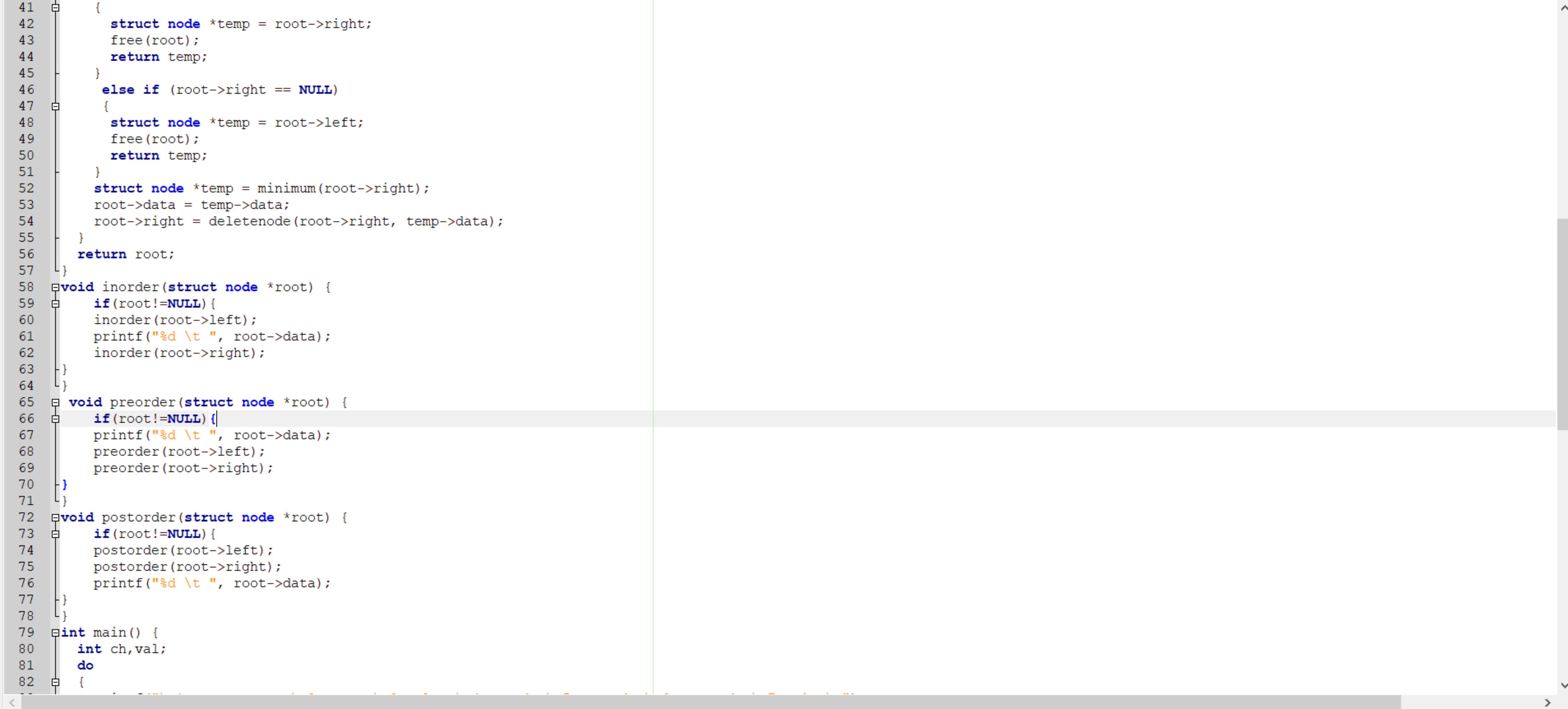
```
1.Create Root
2.Insert
3.Delete
4.Inorder
5.Preorder
6.Postorder
7.Exit

Enter your option:5

Preorder traversal:
 25      16      18      36      30      27
1.Create Root
2.Insert
3.Delete
4.Inorder
5.Preorder
6.Postorder
7.Exit

Enter your option:6

Postorder traversal:
  18     16      27      30      36      25
1.Create Root
2.Insert
3.Delete
4.Inorder
5.Preorder
6.Postorder
7.Exit

Enter your option:3
Enter a data to delete   : 27

1.Create Root
2.Insert
3.Delete
4.Inorder
5.Preorder
6.Postorder
7.Exit
```



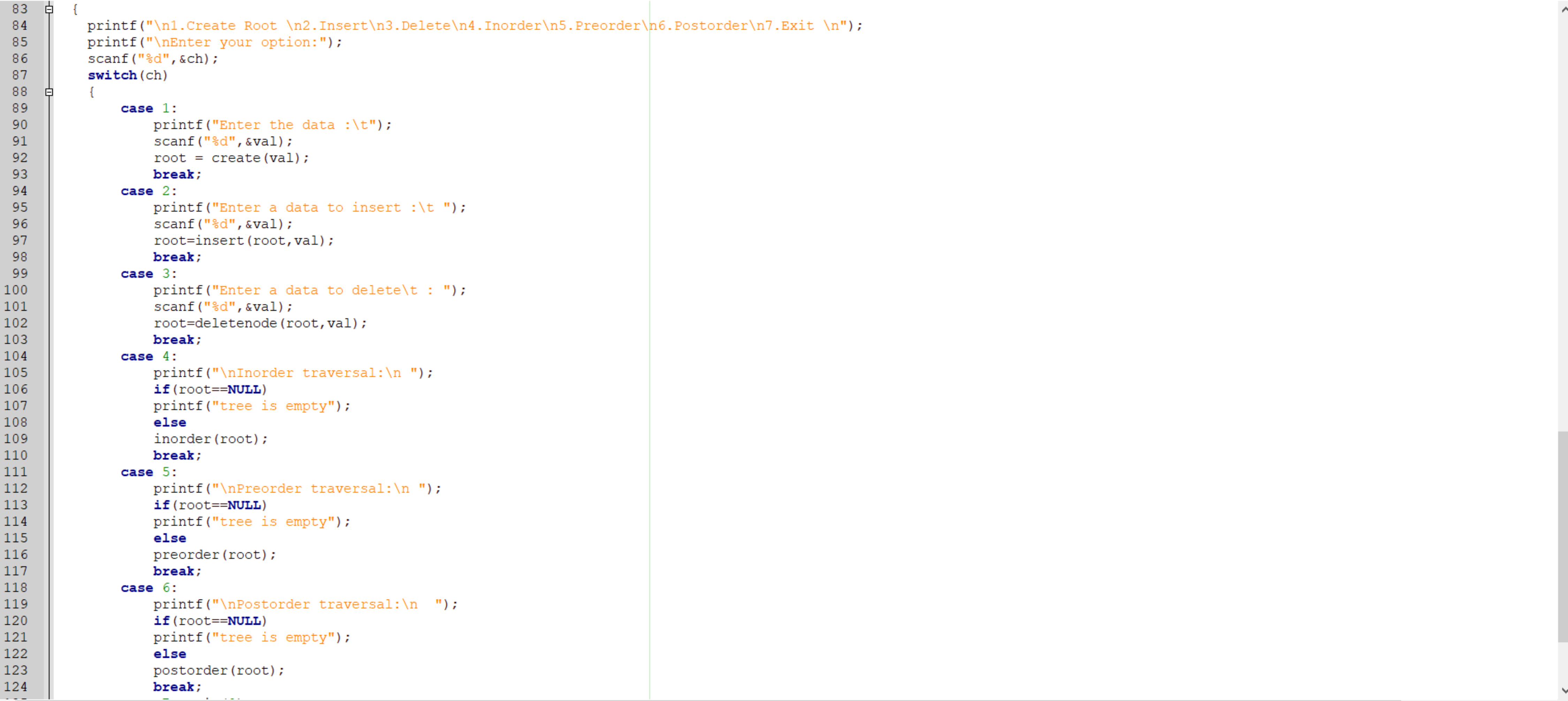
```
1.Create Root
2.Insert
3.Delete
4.Inorder
5.Preorder
6.Postorder
7.Exit

Enter your option:4

Inorder traversal:
 16      18      25      30      36
1.Create Root
2.Insert
3.Delete
4.Inorder
5.Preorder
6.Postorder
7.Exit

Enter your option:5

Preorder traversal:
 25      16      18      36      30
1.Create Root
2.Insert
3.Delete
4.Inorder
5.Preorder
6.Postorder
7.Exit

Enter your option:6

Postorder traversal:
  18     16      30      36      25
1.Create Root
2.Insert
3.Delete
4.Inorder
5.Preorder
6.Postorder
7.Exit
```



```c
#include <stdio.h>
#include <stdlib.h>
struct node {
  int data;
  struct node *left, *right;
};
struct node *root = NULL;
struct node *create(int item)
{
  struct node *temp = (struct node *)malloc(sizeof(struct node));
  temp->data = item;
  temp->left = temp->right = NULL;
  return temp;
}
struct node *insert(struct node *node, int val)
 {
  if (node == NULL)
  return create(val);
  if (val < node->data)
    node->left = insert(node->left, val);
  else
    node->right = insert(node->right, val);
  return node;
}
struct node *minimum(struct node *node)
{
  struct node *current = node;
  while (current && current->left != NULL)
    current = current->left;
  return current;
}
struct node *deletenode(struct node *root, int val) {
  if (root == NULL)
  return root;
  if (val < root->data)
    root->left = deletenode(root->left, val);
  else if (val > root->data)
    root->right = deletenode(root->right, val);
  else {
    if (root->left == NULL)
    {
      struct node *temp = root->right;
```



```c
    {
      struct node *temp = root->right;
      free(root);
      return temp;
    }
     else if (root->right == NULL)
     {
      struct node *temp = root->left;
      free(root);
      return temp;
    }
    struct node *temp = minimum(root->right);
    root->data = temp->data;
    root->right = deletenode(root->right, temp->data);
  }
  return root;
}
void inorder(struct node *root) {
    if(root!=NULL){
    inorder(root->left);
    printf("%d \t ", root->data);
    inorder(root->right);
}
}
 void preorder(struct node *root) {
    if(root!=NULL){
    printf("%d \t ", root->data);
    preorder(root->left);
    preorder(root->right);
}
}
void postorder(struct node *root) {
    if(root!=NULL){
    postorder(root->left);
    postorder(root->right);
    printf("%d \t ", root->data);
}
}
int main() {
  int ch,val;
  do
  {
```



```
{
  printf("\n1.Create Root \n2.Insert\n3.Delete\n4.Inorder\n5.Preorder\n6.Postorder\n7.Exit \n");
  printf("\nEnter your option:");
  scanf("%d",&ch);
  switch(ch)
  {
      case 1:
          printf("Enter the data :\t");
          scanf("%d",&val);
          root = create(val);
          break;
      case 2:
          printf("Enter a data to insert :\t ");
          scanf("%d",&val);
          root=insert(root,val);
          break;
      case 3:
          printf("Enter a data to delete\t : ");
          scanf("%d",&val);
          root=deletenode(root,val);
          break;
      case 4:
          printf("\nInorder traversal:\n ");
          if(root==NULL)
          printf("tree is empty");
          else
          inorder(root);
          break;
      case 5:
          printf("\nPreorder traversal:\n ");
          if(root==NULL)
          printf("tree is empty");
          else
          preorder(root);
          break;
      case 6:
          printf("\nPostorder traversal:\n  ");
          if(root==NULL)
          printf("tree is empty");
          else
          postorder(root);
          break;
```

```
            scanf("%d",&val);
            root = create(val);
            break;
        case 2:
            printf("Enter a data to insert :\t ");
            scanf("%d",&val);
            root=insert(root,val);
            break;
        case 3:
            printf("Enter a data to delete\t : ");
            scanf("%d",&val);
            root=deletenode(root,val);
            break;
        case 4:
            printf("\nInorder traversal:\n ");
            if(root==NULL)
            printf("tree is empty");
            else
            inorder(root);
            break;
        case 5:
            printf("\nPreorder traversal:\n ");
            if(root==NULL)
            printf("tree is empty");
            else
            preorder(root);
            break;
        case 6:
            printf("\nPostorder traversal:\n  ");
            if(root==NULL)
            printf("tree is empty");
            else
            postorder(root);
            break;
        case 7: exit(0);
        default :
            printf("Error");
    }
  }while(ch!=7);
  return 0;
}
```