# DAY 3

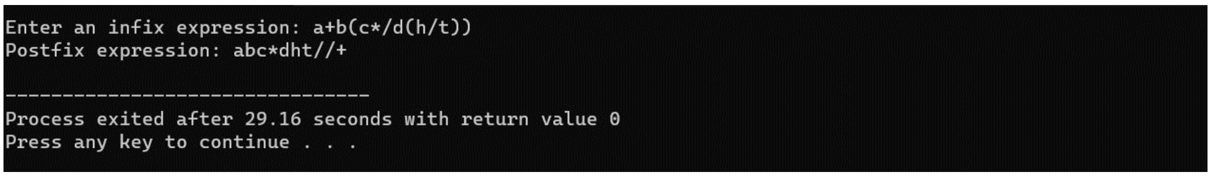
1.Write a c program for infix to postfix expression?

**Program:**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX_SIZE 100
struct Stack {
   char items[MAX_SIZE];
   int top;
};

void initialize(struct Stack *stack) {
   stack->top = -1;
}
int isEmpty(struct Stack *stack) {
   return stack->top == -1;
}
void push(struct Stack *stack, char item) {
   if (stack->top >= MAX_SIZE - 1) {
      printf("Stack is full. Cannot push.\n");
      return;
   }
   stack->items[++stack->top] = item;
}
char pop(struct Stack *stack) {
   if (isEmpty(stack)) {
      printf("Stack is empty. Cannot pop.\n");
      return '\0';
   }
   return stack->items[stack->top--];
```

```
}
int isOperator(char ch) {
    return ch == '+' || ch == '-' || ch == '*' || ch == '/';
}
int precedence(char ch) {
    if (ch == '+' || ch == '-')
        return 1;
    if (ch == '*' || ch == '/')
        return 2;
    return 0;
}
void infixToPostfix(char infix[], char postfix[]) {
    struct Stack stack;
    initialize(&stack);
    int postfixIndex = 0, i;
    for ( i = 0; infix[i] != '\0'; i++) {
        char ch = infix[i];
        if (ch == ' ')
            continue;
        if (isdigit(ch) || isalpha(ch)) {
            postfix[postfixIndex++] = ch;
        } else if (ch == '(') {
            push(&stack, ch);
        } else if (ch == ')') {
            while (!isEmpty(&stack) && stack.items[stack.top] != '(') {
                postfix[postfixIndex++] = pop(&stack);
            }
            pop(&stack);
        } else if (isOperator(ch)) {
            while (!isEmpty(&stack) && precedence(stack.items[stack.top]) >= precedence(ch)) {
                postfix[postfixIndex++] = pop(&stack);
            }
            push(&stack, ch);
        }
    }
    while (!isEmpty(&stack)) {
        postfix[postfixIndex++] = pop(&stack);
    }
    postfix[postfixIndex] = '\0';
}
int main() {
    char infix[MAX_SIZE], postfix[MAX_SIZE];
    printf("Enter an infix expression: ");
    gets(infix);
    infixToPostfix(infix, postfix);
    printf("Postfix expression: %s\n", postfix);
    return 0;
}
```

**Output:**



```
Enter an infix expression: a+b(c*/d(h/t))
Postfix expression: abc*dht//+

--------------------------------
Process exited after 29.16 seconds with return value 0
Press any key to continue . . .
```

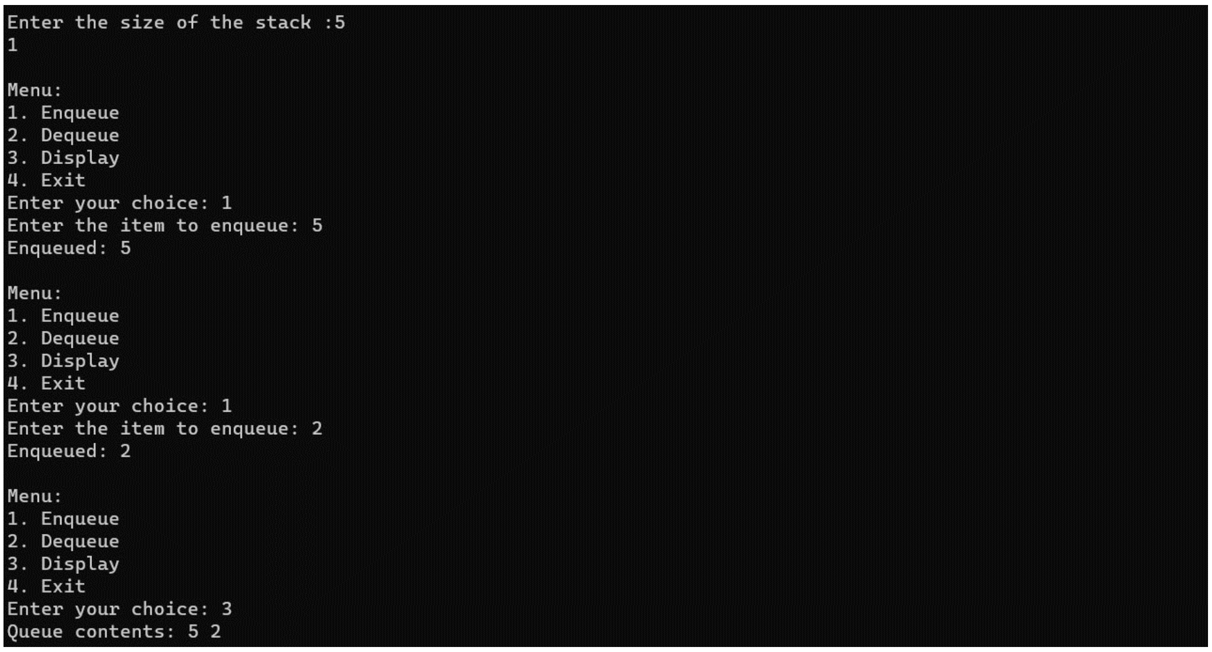
2.Write a c program for queue data structure?

**Program:**

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_SIZE 100
struct Queue {
    int items[MAX_SIZE];
    int front;
    int rear;
};
```

```
void initialize(struct Queue *queue) {
    queue->front = -1;
    queue->rear = -1;
}
int isEmpty(struct Queue *queue) {
    return queue->front == -1;
}
int isFull(struct Queue *queue) {
    return (queue->rear + 1) % MAX_SIZE == queue->front;
}
void enqueue(struct Queue *queue, int item) {
    if (isFull(queue)) {
        printf("Queue is full. Cannot enqueue %d.\n", item);
        return;
    }
    if (isEmpty(queue)) {
        queue->front = 0;
        queue->rear = 0;
    } else {
        queue->rear = (queue->rear + 1) % MAX_SIZE;
    }
    queue->items[queue->rear] = item;
    printf("Enqueued: %d\n", item);
}
int dequeue(struct Queue *queue) {
    if (isEmpty(queue)) {
        printf("Queue is empty. Cannot dequeue.\n");
        return -1;
    }
    int dequeuedItem = queue->items[queue->front];
    if (queue->front == queue->rear) {
        queue->front = -1;
        queue->rear = -1;
    } else {
        queue->front = (queue->front + 1) % MAX_SIZE;
    }
    printf("Dequeued: %d\n", dequeuedItem);
    return dequeuedItem;
}
void display(struct Queue *queue) {
    if (isEmpty(queue)) {
        printf("Queue is empty.\n");
        return;
    }
    printf("Queue contents:");
    int i = queue->front;
    while (i != queue->rear) {
        printf(" %d", queue->items[i]);
        i = (i + 1) % MAX_SIZE;
    }
    printf(" %d", queue->items[i]);
    printf("\n");
}
int main() {
    struct Queue queue;
    initialize(&queue);
    int choice, item, n, i;
    printf("Enter the size of the stack :");
    scanf("%d",&n);
    for(i = 0;i < n;i++){
            scanf("%d",&n);
            }
    do {
        printf("\nMenu:\n");
        printf("1. Enqueue\n");
        printf("2. Dequeue\n");
        printf("3. Display\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter the item to enqueue: ");
                scanf("%d", &item);
                enqueue(&queue, item);
                break;
            case 2:
                dequeue(&queue);
                break;
            case 3:
                display(&queue);
                break;
            case 4:
                printf("Exiting...\n");
                break;
            default:
                printf("Invalid choice!\n");
        }
    } while (choice != 4);
    return 0;
}
```

**Output:**

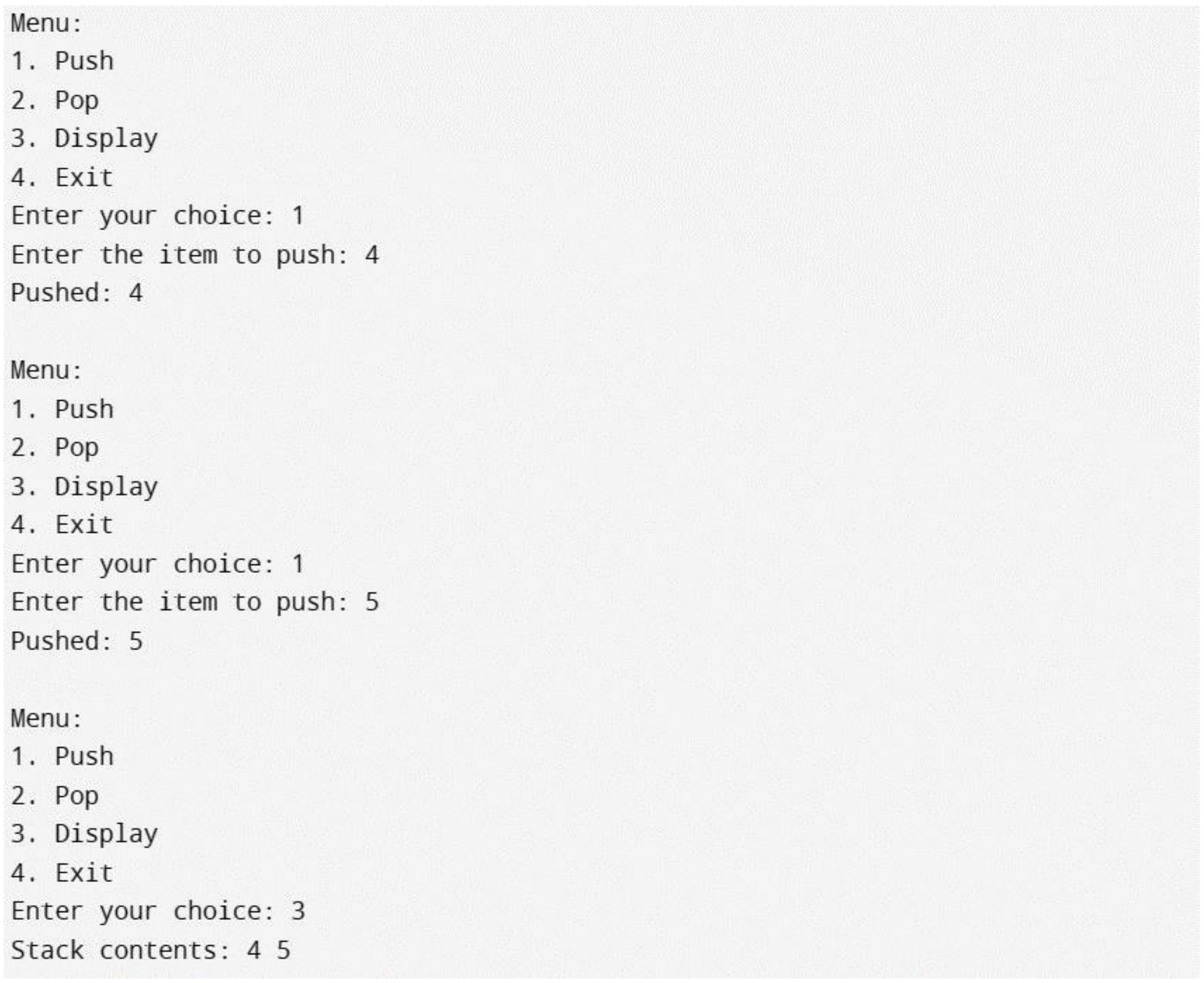

```
Enter the size of the stack :5
1

Menu:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 1
Enter the item to enqueue: 5
Enqueued: 5

Menu:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 1
Enter the item to enqueue: 2
Enqueued: 2

Menu:
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter your choice: 3
Queue contents: 5 2
```

3.Write a c program for to implement stack operations?

**Program:**

```
#include <stdio.h>

#include <stdlib.h>

#define MAX_SIZE 100

struct Stack {

   int items[MAX_SIZE];

   int top;

};

void initialize(struct Stack *stack) {

   stack->top = -1;

}

int isEmpty(struct Stack *stack) {

   return stack->top == -1;

}

int isFull(struct Stack *stack) {

   return stack->top == MAX_SIZE - 1;

}

void push(struct Stack *stack, int item) {

   if (isFull(stack)) {
       printf("Stack is full. Cannot push %d.\n", item);
       return;
   }
   stack->items[++stack->top] = item;
   printf("Pushed: %d\n", item);
}
int pop(struct Stack *stack) {
   if (isEmpty(stack)) {
       printf("Stack is empty. Cannot pop.\n");
       return -1;
   }
   int poppedItem = stack->items[stack->top--];
   printf("Popped: %d\n", poppedItem);
   return poppedItem;
}
void display(struct Stack *stack) {
   if (isEmpty(stack)) {
       printf("Stack is empty.\n");
       return;
   }
   printf("Stack contents:");
   for (int i = 0; i <= stack->top; i++) {
       printf(" %d", stack->items[i]);
   }
   printf("\n");
}
int main() {
   struct Stack stack;
   initialize(&stack);
   int choice, item;
   do {
       printf("\nMenu:\n");
       printf("1. Push\n");
       printf("2. Pop\n");
       printf("3. Display\n");
       printf("4. Exit\n");
       printf("Enter your choice: ");
       scanf("%d", &choice);
       switch (choice) {
           case 1:
               printf("Enter the item to push: ");
               scanf("%d", &item);
               push(&stack, item);
               break;
           case 2:
               pop(&stack);
               break;
           case 3:
               display(&stack);
               break;
           case 4:
               printf("Exiting...\n");
               break;
           default:
               printf("Invalid choice!\n");
       }
   } while (choice != 4);
   return 0;
}
```

**Output:**



```
Menu:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 1
Enter the item to push: 4
Pushed: 4

Menu:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 1
Enter the item to push: 5
Pushed: 5

Menu:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 3
Stack contents: 4 5
```

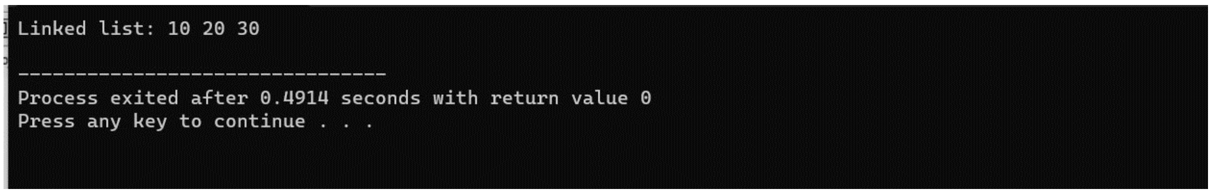
4.Write a c program to implement linked list?

**Program :**

```
#include <stdio.h>

#include <stdlib.h>

struct Node {

   int data;

   struct Node* next;

};

struct Node* createNode(int data) {

   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));

   newNode->data = data;

   newNode->next = NULL;

   return newNode;

}
```

```c
void insertEnd(struct Node** head, int data) {
   struct Node* newNode = createNode(data);
   if (*head == NULL) {
      *head = newNode;
      return;
   }
   struct Node* current = *head;
   while (current->next != NULL) {
      current = current->next;
   }
   current->next = newNode;
}
void displayList(struct Node* head) {
   struct Node* current = head;
   while (current != NULL) {
      printf("%d ", current->data);
      current = current->next;
   }
   printf("\n");
}
int main() {
   struct Node* head = NULL;
   insertEnd(&head, 10);
   insertEnd(&head, 20);
   insertEnd(&head, 30);
   printf("Linked list: ");
   displayList(head);
   return 0;
}
```

**Output:**



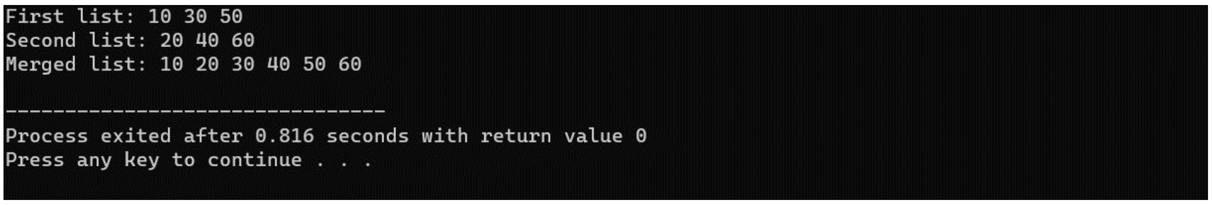
```
Linked list: 10 20 30

--------------------------------
Process exited after 0.4914 seconds with return value 0
Press any key to continue . . .
```

5.Write a c program for merge two list?

**Program:**

```
#include <stdio.h>

#include <stdlib.h>

struct Node {

   int data;

   struct Node* next;

};

struct Node* createNode(int data) {

   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));

   newNode->data = data;

   newNode->next = NULL;

   return newNode;

}

void insertEnd(struct Node** head, int data) {

   struct Node* newNode = createNode(data);

   if (*head == NULL) {

      *head = newNode;

      return;

   }

   struct Node* current = *head;

   while (current->next != NULL) {

      current = current->next;

   }

   current->next = newNode;

}

struct Node* mergeLists(struct Node* list1, struct Node* list2) {
    if (list1 == NULL) return list2;
    if (list2 == NULL) return list1;
    struct Node* result = NULL;
    if (list1->data <= list2->data) {
        result = list1;
        result->next = mergeLists(list1->next, list2);
    } else {
        result = list2;
        result->next = mergeLists(list1, list2->next);
    }
    return result;
}
void displayList(struct Node* head) {
    struct Node* current = head;
    while (current != NULL) {
        printf("%d ", current->data);
        current = current->next;
    }
    printf("\n");
}
int main() {
    struct Node* list1 = NULL;
    struct Node* list2 = NULL;
    insertEnd(&list1, 10);
    insertEnd(&list1, 30);
    insertEnd(&list1, 50);
    insertEnd(&list2, 20);
    insertEnd(&list2, 40);
    insertEnd(&list2, 60);
    printf("First list: ");
    displayList(list1);
    printf("Second list: ");
    displayList(list2);
    struct Node* mergedList = mergeLists(list1, list2);
    printf("Merged list: ");
    displayList(mergedList);

    return 0;
}
```

**Output:**



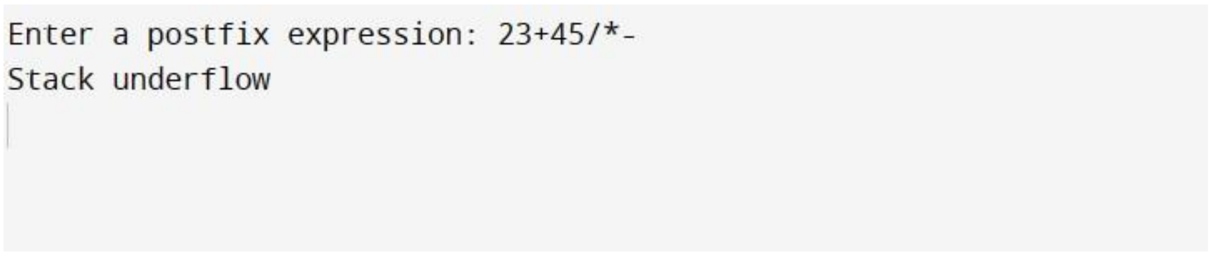
```
First list: 10 30 50
Second list: 20 40 60
Merged list: 10 20 30 40 50 60

--------------------------------
Process exited after 0.816 seconds with return value 0
Press any key to continue . . .
```

6.Write a c program to evaluate the postfix expression?

**Program:**

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>
#define MAX_STACK_SIZE 100
typedef struct {
    int data[MAX_STACK_SIZE];
    int top;
} Stack;
void initialize(Stack *s) {
    s->top = -1;
}
void push(Stack *s, int value) {
    if (s->top < MAX_STACK_SIZE - 1) {
        s->top++;
        s->data[s->top] = value;
    } else {
        printf("Stack overflow\n");
        exit(1);
    }
}
int pop(Stack *s) {
    if (s->top >= 0) {
        int value = s->data[s->top];
        s->top--;
        return value;
    } else {
        printf("Stack underflow\n");
        exit(1);
    }
}
int evaluatePostfix(char postfix[]) {
    Stack stack;
    initialize(&stack);
    for (int i = 0; postfix[i] != '\0'; i++) {
        if (isdigit(postfix[i])) {
            push(&stack, postfix[i] - '0');
        } else {
            int operand2 = pop(&stack);
            int operand1 = pop(&stack);
            switch (postfix[i]) {
                case '+':
                    push(&stack, operand1 + operand2);
                    break;
                case '-':
                    push(&stack, operand1 - operand2);
                    break;
                case '*':
                    push(&stack, operand1 * operand2);
                    break;
                case '/':
                    push(&stack, operand1 / operand2);
                    break;
                default:
                    printf("Invalid operator\n");
                    exit(1);
            }
        }
    }
    return pop(&stack);
}
int main() {
    char postfix[100];
    printf("Enter a postfix expression: ");
    scanf("%s", postfix);
    int result = evaluatePostfix(postfix);
    printf("Result: %d\n", result);
    return 0;
}
```

**Output:**



```
Enter a postfix expression: 23+45/*-
Stack underflow
```

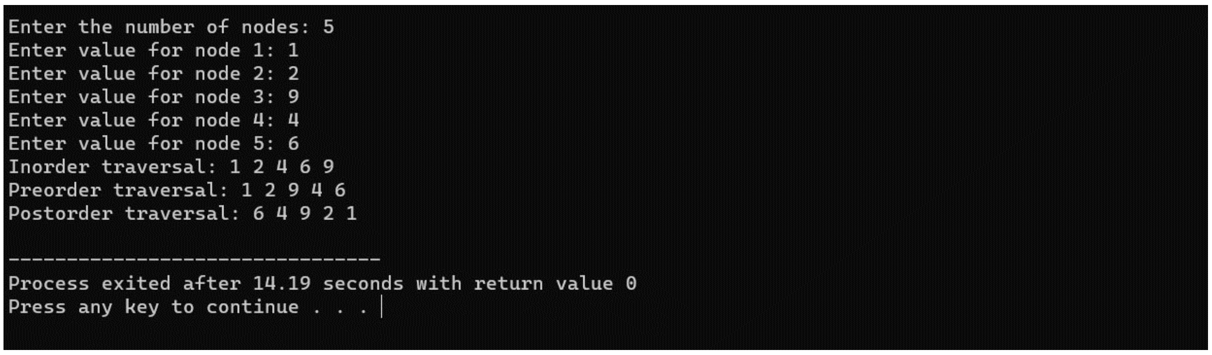
7.write a c program to implement tree traversals?

**Program:**

```
#include <stdio.h>

#include <stdlib.h>

struct Node {

   int data;

   struct Node *left;

   struct Node *right;

};

struct Node *newNode(int data) {

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

   node->data = data;

   node->left = NULL;

   node->right = NULL;

   return node;

}

void inorderTraversal(struct Node *root) {

   if (root != NULL) {

      inorderTraversal(root->left);

      printf("%d ", root->data);

      inorderTraversal(root->right);

   }

}

void preorderTraversal(struct Node *root) {

   if (root != NULL) {

      printf("%d ", root->data);

      preorderTraversal(root->left);

      preorderTraversal(root->right);

   }

}

void postorderTraversal(struct Node *root) {

   if (root != NULL) {

      postorderTraversal(root->left);
```

```
        postorderTraversal(root->right);
        printf("%d ", root->data);
    }
}
int main() {
    struct Node *root = NULL;
    int n, i;
    printf("Enter the number of nodes: ");
    scanf("%d", &n);
    for ( i = 0; i < n; i++) {
        int value;
        printf("Enter value for node %d: ", i + 1);
        scanf("%d", &value);
        if (root == NULL) {
            root = newNode(value);
        } else {
            struct Node *current = root;
            struct Node *parent = NULL;
            while (current != NULL) {
                parent = current;
                if (value < current->data) {
                    current = current->left;
                } else {
                    current = current->right;
                }
            }
            if (value < parent->data) {
                parent->left = newNode(value);
            } else {
                parent->right = newNode(value);
            }
        }
    }
    printf("Inorder traversal: ");
    inorderTraversal(root);
    printf("\n");
    printf("Preorder traversal: ");
    preorderTraversal(root);
    printf("\n");
    printf("Postorder traversal: ");
    postorderTraversal(root);
    printf("\n");
    return 0;
}
```

**Output:**



```
Enter the number of nodes: 5
Enter value for node 1: 1
Enter value for node 2: 2
Enter value for node 3: 9
Enter value for node 4: 4
Enter value for node 5: 6
Inorder traversal: 1 2 4 6 9
Preorder traversal: 1 2 9 4 6
Postorder traversal: 6 4 9 2 1

--------------------------------
Process exited after 14.19 seconds with return value 0
Press any key to continue . . .
```