BST:

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

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

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

struct Node* insert(struct Node *root, int value) {
    if (root == NULL)
        return createNode(value);

    if (value < root->data)
        root->left = insert(root->left, value);
    else if (value > root->data)
        root->right = insert(root->right, value);

    return root;
}
```

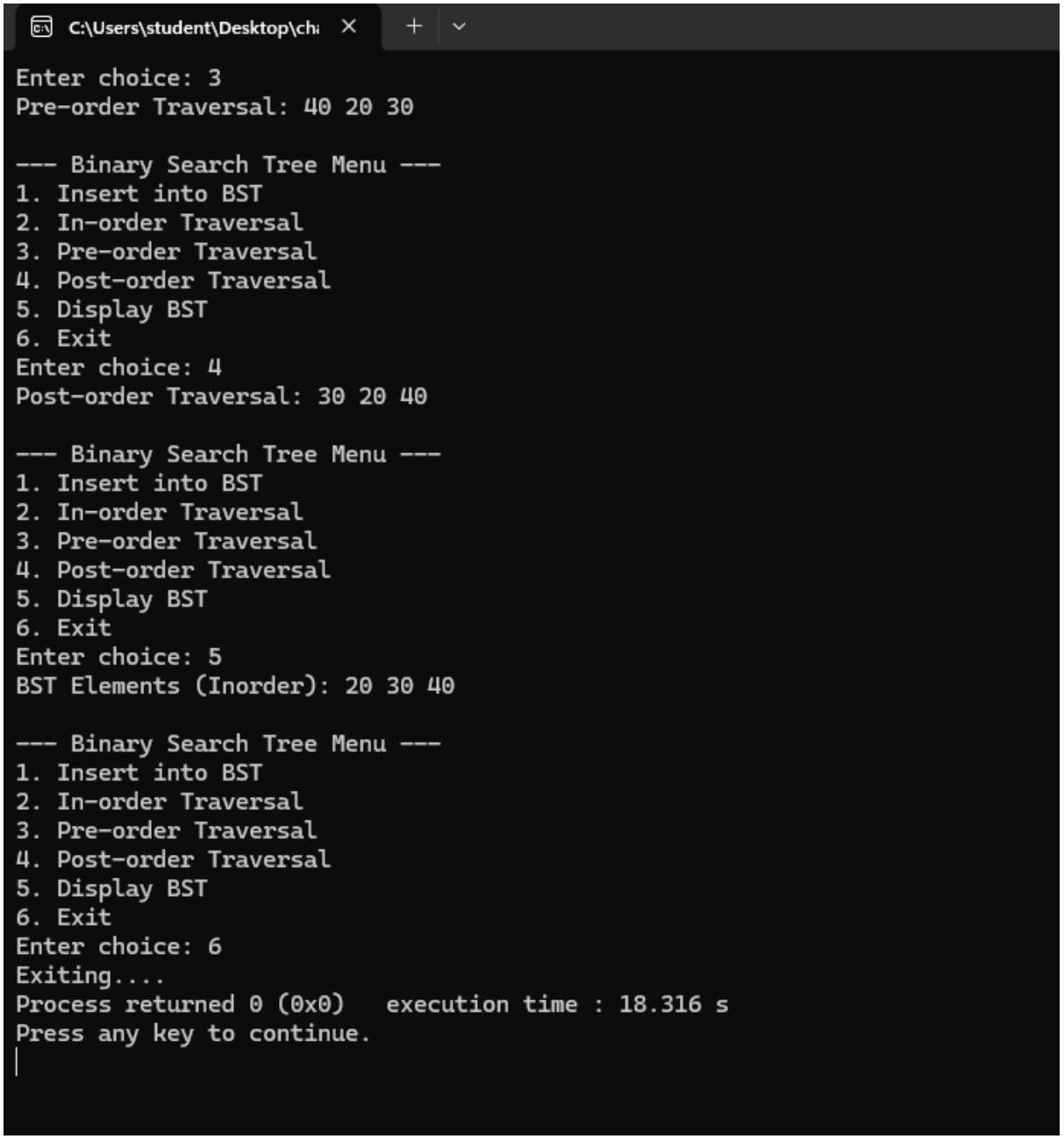
```
/* Inorder Traversal: Left -> Root -> Right */
void inorder(struct Node *root) {
   if (root == NULL)
      return;
   inorder(root->left);
   printf("%d ", root->data);
   inorder(root->right);
}

/* Preorder Traversal: Root -> Left -> Right */
void preorder(struct Node *root) {
   if (root == NULL)
      return;
   printf("%d ", root->data);
   preorder(root->left);
   preorder(root->right);
}

/* Postorder Traversal: Left -> Right -> Root */
void postorder(struct Node *root) {
   if (root == NULL)
      return;
   postorder(root->left);
   postorder(root->right);
   printf("%d ", root->data);
}

void display(struct Node *root) {
```

```
    printf("BST Elements (Inorder): ");
    inorder(root);
    printf("\n");
}

int main() {
    struct Node *root = NULL;
    int choice, value;

    while (1) {
        printf("\n--- Binary Search Tree Menu ---\n");
        printf("1. Insert into BST\n");
        printf("2. In-order Traversal\n");
        printf("3. Pre-order Traversal\n");
        printf("4. Post-order Traversal\n");
        printf("5. Display BST\n");
        printf("6. Exit\n");
        printf("Enter choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter value to insert: ");
                scanf("%d", &value);
                root = insert(root, value);
                break;

            case 2:
                printf("In-order Traversal: ");
```

```
            inorder(root);
            printf("\n");
            break;

        case 3:
            printf("Pre-order Traversal: ");
            preorder(root);
            printf("\n");
            break;

        case 4:
            printf("Post-order Traversal: ");
            postorder(root);
            printf("\n");
            break;

        case 5:
            display(root);
            break;

        case 6:
            printf("Exiting....");
            exit(0);

        default:
            printf("Invalid choice! Try again.\n");
    }
}
```

```
    return 0;
}
```

OUTPUT:



OBSERVATION:

Page No:
Date.

8/12/25 Prg-8a.

WAP a.) to construct a binary Search tree
b.) To traverse the tree using all the methods i.e, in-order, pre-order and post-order
c.) To display the elements in the tree

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

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

struct Node * CreateNode (int value) {
    struct Node *newNode = (struct Node *)
                malloc (sizeof (struct
                Node));
    newNode -> data = value;
    newNode -> left = newNode -> right = NULL;
    return newNode;
}

struct Node * insert (struct Node *root,
                int value) {
    if (root == NULL)
        return createNode (value);

    if (value < root -> data)
        root -> left = insert (root -> left,
                value);
```

```
else if (value > root->data)
    root->right = insert(root->right,
                         value);
    return root;
}

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

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

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

```
void display (struct Node * root) {
    printf ("BST elements (Inorder): ");
    inorder (root);
    printf ("\n");
}

void int main () {
    struct Node *root = NULL;
    int choice, value;

while (1) {
    printf ("\n--- Binary Search Tree Menu ---\n");
    printf ("1. Insert into BST\n");
    printf ("2. In-order Traversal\n");
    printf ("3. Pre-order Traversal\n");
    printf ("4. Post-order Traversal\n");
    printf ("5. Display BST\n");
    printf ("6. Exit\n");
    printf ("Enter choice: ");
    pscanf ("%d", &choice);

    switch (choice) {
        case 1:
            printf ("Enter value to
                    insert : ");
            scanf ("%d", &value);
            root = insert (root, value);
            break;

        case 2:
            printf ("In-order Traversal: ");
```

```
            inorder(root);
            printf("\n");
            break;

        case 3:
            printf("Pre-order Traversal: ");
            preorder(root);
            printf("\n");
            break;

        case 4:
            printf("Post-order Traversal: ");
            postorder(root);
            printf("\n");
            break;

        case 5:
            display(root);
            break;

        case 6:
            printf("Exiting ...\n");
            exit(0);

        default:
            printf("Invaild choice! Try
                    again.\n");
        }
    }

    return 0;
}
```

Date.

Output:

---- Binary Search Tree Menu ----

1. Insert into BST
2. In-order Traversal
3. Pre-order Traversal
4. Post-order Traversal
5. Display BST
6. Exit.

Enter your choice : 1
Enter value to insert : 40

Enter your choice : 1
Enter value to insert : 20

Enter your choice : 1
Enter value to insert : 30

Enter your choice : 2
In-order Traversal : 20 30 40

Enter your choice : 3
Pre-order Traversal : 40 20 30

8/4/25

Enter your choice : 4
Post-order Traversal : 30 20 40

Enter your choice : 5
BST Elements (Inorder) : 20 30 40

Enter your choice : 6
Exiting...