

1st question

```
#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 printPostorder(struct node* node)
{
    if (node == NULL)
        return;
    printPostorder(node->left);
    printPostorder(node->right);
    printf("%d ", node->data);
}
void printInorder(struct node* node)
{
    if (node == NULL)
        return;
    printInorder(node->left);
    printf("%d ", node->data);
    printInorder(node->right);
}
void printPreorder(struct node* node)
{
    if (node == NULL)
        return;
    printf("%d ", node->data);
    printPreorder(node->left);
    printPreorder(node->right);
}
int main()
```

```

{
    struct node *root    = newNode(5);
    root->left           = newNode(4);
    root->right          = newNode(3);
    root->left->left      = newNode(2);
    root->left->right     = newNode(1);
    printf("\nPreorder traversal of binary tree is \n");
    printPreorder(root);
    printf("\nInorder traversal of binary tree is \n");
    printInorder(root);
    printf("\nPostorder traversal of binary tree is \n");
    printPostorder(root);
    return 0;
}

```

2nd question

```

/* program to construct tree from inorder traversal */
#include<stdio.h>
#include<stdlib.h>

/* A binary tree node has data, pointer to left child
   and a pointer to right child */
struct node
{
    int data;
    struct node* left;
    struct node* right;
};

/* Prototypes of a utility function to get the maximum
   value in inorder[start..end] */
int max(int inorder[], int strt, int end);

/* A utility function to allocate memory for a node */
struct node* newNode(int data);

/* Recursive function to construct binary of size len from
   Inorder traversal inorder[]. Initial values of start and end
   should be 0 and len -1. */
struct node* buildTree (int inorder[], int start, int end)
{

```

```

    if (start > end)
        return NULL;

    /* Find index of the maximum element from Binary Tree */
    int i = max (inorder, start, end);

    /* Pick the maximum value and make it root */
    struct node *root = newNode(inorder[i]);

    /* If this is the only element in inorder[start..end],
       then return it */
    if (start == end)
        return root;

    /* Using index in Inorder traversal, construct left and
       right subtress */
    root->left = buildTree (inorder, start, i-1);
    root->right = buildTree (inorder, i+1, end);

    return root;
}

/* UTILITY FUNCTIONS */
/* Function to find index of the maximum value in arr[start...end] */
int max (int arr[], int strt, int end)
{
    int i, max = arr[strt], maxind = strt;
    for(i = strt+1; i <= end; i++)
    {
        if(arr[i] > max)
        {
            max = arr[i];
            maxind = i;
        }
    }
    return maxind;
}

/* Helper function that allocates a new node with the
   given data and NULL left and right pointers. */
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;
}

/* This funtcion is here just to test buildTree() */
void printInorder (struct node* node)
{
    if (node == NULL)
        return;

    /* first recur on left child */
    printInorder (node->left);

    /* then print the data of node */
    printf("%d ", node->data);

    /* now recur on right child */
    printInorder (node->right);
}

/* Driver program to test above functions */
int main()
{
    /* Assume that inorder traversal of following tree is given
        40
       / \
      10  30
     /   \
    5     28 */

    int inorder[] = {5, 10, 40, 30, 28};
    int len = sizeof(inorder)/sizeof(inorder[0]);
    struct node *root = buildTree(inorder, 0, len - 1);

    /* Let us test the built tree by printing Insorder traversal */
    printf("\n Inorder traversal of the constructed tree is \n");
    printInorder(root);
    return 0;
}

```

3rd question

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

```
int main()
{
    int array[100], find, a, n;
    printf("Enter number of elements in array\n");
    scanf("%d", &n);
    printf("Enter %d integer(s)\n", n);
    for (a = 0; a < n; a++)
    {
        scanf("%d", &array[a]);
        printf("Enter a number to search\n");
    }
    scanf("%d", &find);
    for (c = 0; c < n; c++)
    {
        if (array[a] == find) /* If required element is found */
        {
            printf("%d is present at location %d.\n", find, a+1);
            break;
        }
    }
    if (a == n)
        printf("%d isn't present in the array.\n", search);
    return 0;
}
```

4th question

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

```
int main()
{
    int i, front, last, middle, n, search, array[100];
    printf("Enter number of elements\n");
    scanf("%d", &n);
    printf("Enter %d integers\n", n);
    for (i = 0; i < n; i++)
        scanf("%d", &array[i]);
    printf("Enter value to find\n");
    scanf("%d", &search);
}
```

```
front = 0;
last = n - 1;
middle = (front + last)/2;
while (front <= last) {
    if (array[middle] < search)
        front = middle + 1;
    else if (array[middle] == search) {
        printf("%d found at location %d.\n", search, middle+1);
        break;
    }
    else
        end = middle - 1;
    middle = (front + last)/2;
}
if (front > last)
    printf("Not present: %d isn't present in the list.\n", search);
return 0;
}
```