3)/*Write a C program for linear search algorithm*/

```
#include <stdio.h>
int main()
{
  int array[50], search, i, x;
  printf("Enter no.of elements you want to enter in an array: ");
  scanf("%d", &x);
  printf("Enter %d integers: ", x);
  for (i = 0; i < x; i++)
  scanf("%d", &array[i]);
  printf("Enter a number for searching in an array: ");
  scanf("%d", &search);
  for (i = 0; i < x; i++)
  {
    if (array[i] == search)
    {
      printf("%d is present at location %d.\n", search, i);
      break;
    }
  }
  if (i == x)
    printf("%d isn't present in an array.\n", search);
    return 0;
}
```

4)Write a C program for binary search algorithm

```
#include <stdio.h>
int main()
{
  int i, initial, max, middle, x, search, array[50];

  printf("Enter no.of elements you want to enter in the array: ");
  scanf("%d", &x);

  printf("Enter %d integers: ", x);

  for (i = 0; i < x; i++)
    scanf("%d", &array[i]);

  printf("Enter the value for to search: ");
  scanf("%d", &search);

  initial = 0;
  max = x - 1;
```

```
  middle = (initial+max)/2;

  while (initial <= max) {
    if (array[middle] < search)
      initial = middle + 1;
    else if (array[middle] == search) {
      printf("%d found at the location %d.\n", search, middle+1);
      break;
    }
    else
      max = middle - 1;

    middle = (initial + max)/2;
  }
  if (initial > max)
    printf("Invalid Entry %d not present in the list.\n", search);

  return 0;
}
```

1)

```
/*Write a C program to print preorder, inorder, and postorder traversal on Binary Tree.*/
#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 postorder(struct node* node)
{
if (node == NULL)
return;
postorder(node->left);
postorder(node->right);
printf("%d", node->data);
}
```

```
void inorder(struct node* node)
{
if (node == NULL)
return;
inorder(node->left);
printf("%d", node->data);
inorder(node->right);
}
void preorder(struct node* node)
{
if (node == NULL)
return;
printf("%d ", node->data);
preorder(node->left);
preorder(node->right);
}
int main()
{
struct node *root = newNode(8);
root->left = newNode(3);
root->right = newNode(12);
root->left->left = newNode(0);
root->left->right = newNode(7);
printf("Preorder traversal of binary tree are: \n");
preorder(root);
printf("\nInorder traversal of binary tree are: \n");
inorder(root);
printf("\nPostorder traversal of binary tree are: \n");
postorder(root);
getchar();
return 0;
}
```

2)

```
/*Write a C program to create (or insert) and inorder traversal on Binary Search Tree*/
#include<stdio.h>
#include<stdlib.h>
struct btnode
{
 int value;
 struct btnode *rule;
 struct btnode *r;
}*root = NULL, *temp = NULL, *x2, *x1;
int insert();
int inorder(struct btnode *x);
int flag = 1;
```

```
int main()
{
 int ch;
 printf("\n1)Insert an element into the tree: \n2) Inorder Traversal: \n3) Exit: \n");
 while(1)
 {
 printf("\nEnter your choice : ");
 scanf("%d", &ch);
 switch (ch)
 {
 case 1:
 insert();
 break;
 case 2:
 inorder(root);
 break;
 case 3:
 exit(0);
 default :
 printf("Invalid Entry, Enter the choice again: ");
 break;
 }
 }
return 0;
}
int insert()
{
 create();
 if (root == NULL)
 root = temp;
 else
 search(root);
 return 0;
}
int create()
{
 int data;
 printf("Enter the data node to be inserted : ");
 scanf("%d", &data);
 temp = (struct btnode *)malloc(1*sizeof(struct btnode));
 temp->value = data;
 temp->rule = temp->r = NULL;
 return 0;
}
int search(struct btnode *x)
{
```

```
 if ((temp->value > x->value) && (x->r != NULL))
 search(x->r);
 else if ((temp->value > x->value) && (x->r == NULL))
 x->r = temp;
 else if ((temp->value < x->value) && (x->rule != NULL))
 search(x->rule);
 else if ((temp->value < x->value) && (x->rule == NULL))
 x->rule = temp;
 return 0;
}
int inorder(struct btnode *x)
{
 if (root == NULL)
 {
 printf("Sorry! There are No elements to display");
 return;
 }
 if (x->rule != NULL)
 inorder(x->rule);
 printf("%d -> ", x->value);
 if (x->r != NULL)
 inorder(x->r);
 return 0;
}
```