Lab 1:

Question 1 :

*#include* <stdio.h>

*#include* <stdlib.h>

*typedef* *struct* node

{

*struct* node *\**left;

*struct* node *\**right;

*int* val;

} *\** Node;

*void* inorder(Node *tree*)

{

*if* (*tree*)

  {

    inorder(*tree*->left);

    printf("%d\t", *tree*->val);

    inorder(*tree*->right);

  }

}

*void* preorder(Node *tree*)

{

*if* (*tree*)

  {

    printf("%d\t", *tree*->val);

    preorder(*tree*->left);

    preorder(*tree*->right);

  }

}

*void* postorder(Node *tree*)

{

*if* (*tree*)

  {

    postorder(*tree*->left);

    postorder(*tree*->right);

    printf("%d\t", *tree*->val);

  }

}

*void* insert(Node *\*tree*, *int* *val*)

{

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

  n->val *=* *val*;

  n->left *=* NULL;

  n->right *=* NULL;

*if* (*\*tree* *==* NULL)

  {

*\*tree* *=* n;

*return*;

  }

  Node cur *=* *\*tree*;

  Node prev *=* *\*tree*;

*while* (cur)

  {

    prev *=* cur;

*if* ((cur->val) *<* *val*)

    {

      cur *=* cur->right;

    }

*else* *if* ((cur->val) *>* *val*)

    {

      cur *=* cur->left;

    }

  }

*if* (*val* *>* prev->val)

  {

    prev->right *=* n;

  }

*if* (*val* *<* prev->val)

  {

    prev->left *=* n;

  }

}

*void* search(Node *\*tree*, *int* *val*)

{

*if* (*\*tree* *==* NULL)

  {

    printf("\n\*\*Element not found\*\*\n");

    insert(*tree*, *val*);

*return*;

  }

  Node cur *=* *\*tree*;

*while* (cur)

  {

*if* (cur->val *==* *val*)

    {

      printf("\n\*\*Key Found\*\*\n");

*return*;

    }

*else* *if* (cur->val *>* *val*)

    {

      cur *=* cur->left;

    }

*else* *if* (cur->val *<* *val*)

    {

      cur *=* cur->right;

    }

  }

  printf("\n\*\*Element not Found\*\*\n");

  insert(*tree*, *val*);

}

*int* main()

{

*int* choice *=* 0, inp;

  Node tree *=* NULL;

*while* (choice *<* 5)

  {

    printf("1)Search for item\n2)Inorder\n3)Preorder\n4)Postorder\n5)Exit\n");

    printf("Enter your choice : ");

    scanf("%d", *&*choice);

*switch* (choice)

    {

*case* 1:

      printf("Enter the value to be searched : ");

      scanf("%d", *&inp*);

      search(*&tree*, inp);

*break*;

*case* 2:

      printf("\n");

*if* (tree *==* NULL)

      {

        printf("Tree is Empty, nothing to print.\n");

*continue*;

      }

      inorder(tree);

      printf("\n");

*break*;

*case* 3:

      printf("\n");

*if* (tree *==* NULL)

      {

        printf("Tree is Empty, nothing to print.\n");

*continue*;

      }

      preorder(tree);

      printf("\n");

*break*;

*case* 4:

      printf("\n");

*if* (tree *==* NULL)

      {

        printf("Tree is Empty, nothing to print.\n");

*continue*;

      }

      postorder(tree);

      printf("\n");

*break*;

*default*:

*continue*;

    }

  }

*return* 0;

}

Text

Description automatically generated

Graphical user interface, text

Description automatically generated with medium confidence

Graphical user interface, text

Description automatically generated

Question 2 :

*#include* <stdio.h>

*#include* <stdlib.h>

*typedef* *struct* listNode

{

*int* n;

*struct* listNode *\**next;

} *\** ListNode;

*typedef* *struct* list

{

*struct* listNode *\**head;

} *\** List;

*typedef* *struct* graph

{

*int* val;

*struct* list *\**array;

} *\** Graph;

ListNode newListNode(*int* *d*)

{

  ListNode newNode *=* (ListNode)malloc(*sizeof*(*struct* listNode));

  newNode->n *=* *d*;

  newNode->next *=* NULL;

*return* newNode;

}

Graph newGraph(*int* *v*)

{

  Graph newGraph *=* (Graph)malloc(*sizeof*(*struct* graph));

  newGraph->val *=* *v*;

  newGraph->array *=* (List)calloc(*v*, *sizeof*(*struct* list));

*for* (*int* i *=* 0; i *<* *v*; i*++*)

  {

    newGraph->array[i].head *=* NULL;

  }

*return* newGraph;

}

*void* insertEdge(Graph *g*, *int* *first*, *int* *second*)

{

  ListNode newNode *=* newListNode(*second*);

  newNode->next *=* *g*->array[*first*].head;

*g*->array[*first*].head *=* newNode;

  newNode *=* newListNode(*first*);

  newNode->next *=* *g*->array[*second*].head;

*g*->array[*second*].head *=* newNode;

}

*void* displayList(Graph *g*)

{

*for* (*int* i *=* 0; i *<* *g*->val; i*++*)

  {

    ListNode l *=* *g*->array[i].head;

    printf("List of vertex %d\n", i);

*while* (l->next)

    {

      printf("%d -> ", l->n);

      l *=* l->next;

    }

    printf("%d\n", l->n);

  }

}

*void* insertEdgeM(*int* *\*\*matrix*, *int* *first*, *int* *second*)

{

*matrix*[*first*][*second*] *=* 1;

*matrix*[*second*][*first*] *=* 1;

}

*void* displayMatrix(*int* *\*\*matrix*, *int* *n*)

{

*for* (*int* i *=* *-*1; i *<* *n*; i*++*)

  {

*if* (i *!=* *-*1)

      printf("%d -> ", i);

*for* (*int* j *=* 0; j *<* *n*; j*++*)

    {

*if* (i *==* *-*1)

      {

*if* (j *==* 0)

        {

          printf("\t");

        }

        printf("%d\t", j);

*continue*;

      }

*if* (j *==* 0)

      {

        printf("\t");

      }

      printf("%d\t", *matrix*[i][j]);

    }

    printf("\n");

  }

}

*int* main()

{

*int* num *=* 4;

  Graph g *=* newGraph(num);

  insertEdge(g, 1, 3);

  insertEdge(g, 0, 2);

  insertEdge(g, 1, 2);

  insertEdge(g, 2, 3);

  displayList(g);

  printf("\n");

*int* *\*\**matrix *=* (*int* *\*\**)calloc(num, *sizeof*(*int* *\**));

*for* (*int* i *=* 0; i *<* num; i*++*)

  {

    matrix[i] *=* (*int* *\**)calloc(num, *sizeof*(*int*));

*for* (*int* j *=* 0; j *<* num; j*++*)

    {

      matrix[i][j] *=* 0;

    }

  }

  insertEdgeM(matrix, 1, 3);

  insertEdgeM(matrix, 0, 2);

  insertEdgeM(matrix, 1, 2);

  insertEdgeM(matrix, 2, 3);

  displayMatrix(matrix, num);

*return* 0;

}

Text

Description automatically generated