**Objectives**

* To learn about the insertion on linked list.
* To learn about tree traversal.

**Write a program to show the insertion in linked list.**

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

/\* Node Stucture \*/

typedef struct node\_t {

int data;

struct node\_t \*next;

} Node;

/\* Function Declarations \*/

Node \* insert\_top(int, Node \*);

Node \* insert\_bottom(int, Node \*);

Node \* insert\_after(int, int, Node \*);

Node \* insert\_before(int, int, Node \*);

void print(Node \*);

int count(Node \*);

/\* Add a new node to the top of a list \*/

Node \* insert\_top(int num, Node \*head) {

Node \*new\_node;

new\_node = (Node \*) malloc(sizeof(Node));

new\_node->data = num;

new\_node->next= head;

head = new\_node;

return head;

}

/\* Add a new node to the bottom of a list \*/

Node \* insert\_bottom(int num, Node \*head) {

Node \*current\_node = head;

Node \*new\_node;

while ( current\_node != NULL && current\_node->next != NULL) {

current\_node = current\_node->next;

}

new\_node = (Node \*) malloc(sizeof(Node));

new\_node->data = num;

new\_node->next= NULL;

if (current\_node != NULL)

current\_node->next = new\_node;

else

head = new\_node;

return head;}

/\* Add a new node after an element in the list \*/

Node \* insert\_after(int num, int prev\_num, Node \*head) {

Node \*current\_node = head;

Node \*new\_node;

while ( current\_node->data != prev\_num) {

current\_node = current\_node->next;

}

new\_node = (Node \*) malloc(sizeof(Node));

new\_node->data = num;

new\_node->next= current\_node->next;

current\_node->next = new\_node;

return head;

}

/\* Add a new node before an element in the list \*/

Node \* insert\_before(int num, int next\_num, Node \*head) {

Node \*current\_node = head;

Node \*new\_node;

while ( current\_node->next->data != next\_num) {

current\_node = current\_node->next;

}

new\_node = (Node \*) malloc(sizeof(Node));

new\_node->data = num;

new\_node->next= current\_node->next;

current\_node->next = new\_node;

return head;

}

/\* Print all the elements in the linked list \*/

void print(Node \*head) {

Node \*current\_node = head;

while ( current\_node != NULL) {

printf("%d ", current\_node->data);

current\_node = current\_node->next;

}

}

/\* Program main \*/

int main()

{

Node \*head = NULL;

int num, prev\_num, next\_num;

int option;

char \* temp;

char ch;

/\* Display Menu \*/

while(1) {

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

printf(" \* Linked list operations: \*\n");

printf(" \* 1. Insert at the top of list \*\n");

printf(" \* 2. Insert at bottom of list \*\n");

printf(" \* 3. Insert after an element \*\n");

printf(" \* 4. Insert before an element \*\n");

printf(" \* 5. Show all elements \*\n");

printf(" \* 6. Quit \*\n");

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

printf("Choose an option [1-5] : ");

if (scanf("%d", &option) != 1) {

printf(" \*Error: Invalid input. Try again.\n");

scanf("%s", &temp); /\*clear input buffer \*/

continue;

}

switch (option) {

case 1: /\* Add to top\*/

printf(" Enter a number to insert : ");

if (scanf("%d", &num) != 1) {

printf(" \*Error: Invalid input.\n");

scanf("%s", &temp); /\*clear input buffer \*/

continue;

}

head = insert\_top(num, head);

printf("Number %d added to the top of the list", num);

printf("\nPress any key to continue...");

getch();

break;

case 2: /\* add to bottom \*/

printf(" Enter a number to insert : ");

if (scanf("%d", &num) != 1) {

printf(" \*Error: Invalid input. \n");

scanf("%s", &temp);

continue;

}

head = insert\_bottom(num, head);

printf("%d added to the bottom of the list", num);

printf("\nPress any key to continue...");

getch();

break;

case 3: /\* Insert After \*/

printf(" Enter a number to insert : ");

if (scanf("%d", &num) != 1) {

printf(" \*Error: Invalid input.\n");

scanf("%s", &temp);

continue;

}

printf(" After which number do you want to insert : ");

if (scanf("%d", &prev\_num) != 1) {

printf(" \*Error: Invalid input.\n");

scanf("%s", &temp);

continue;

}

if (head != NULL) {

head = insert\_after(num, prev\_num, head);

printf("%d is inserted after %d", num, prev\_num);

}else {

printf("The list is empty", num, prev\_num);

}

printf("\nPress any key to continue...");

getch();

break;

case 4: /\* Insert Before \*/

printf(" Enter a number to insert : ");

if (scanf("%d", &num) != 1) {

printf(" \*Error: Invalid input. \n");

scanf("%s", &temp);

continue;

}

printf(" Before which number do you want to insert : ");

if (scanf("%d", &prev\_num) != 1) {

printf(" \*Error: Invalid input.\n");

scanf("%s", &temp);

continue;

}

if (head != NULL) {

head = insert\_before(num, prev\_num, head);

printf("Number %d inserted before %d", num, prev\_num);

}else {

printf("The list is empty", num, prev\_num);

}

printf("\nPress any key to continue...");

getch();

break;

case 5: /\* Show all elements \*/

printf("\nElements in the list: \n [ ");

print(head);

printf("]\n\nPress any key to continue...");

getch();

break;

case 6: /\* Exit \*/

return(0);

break;

default:

printf("Invalid Option. Please Try again.");

getch();

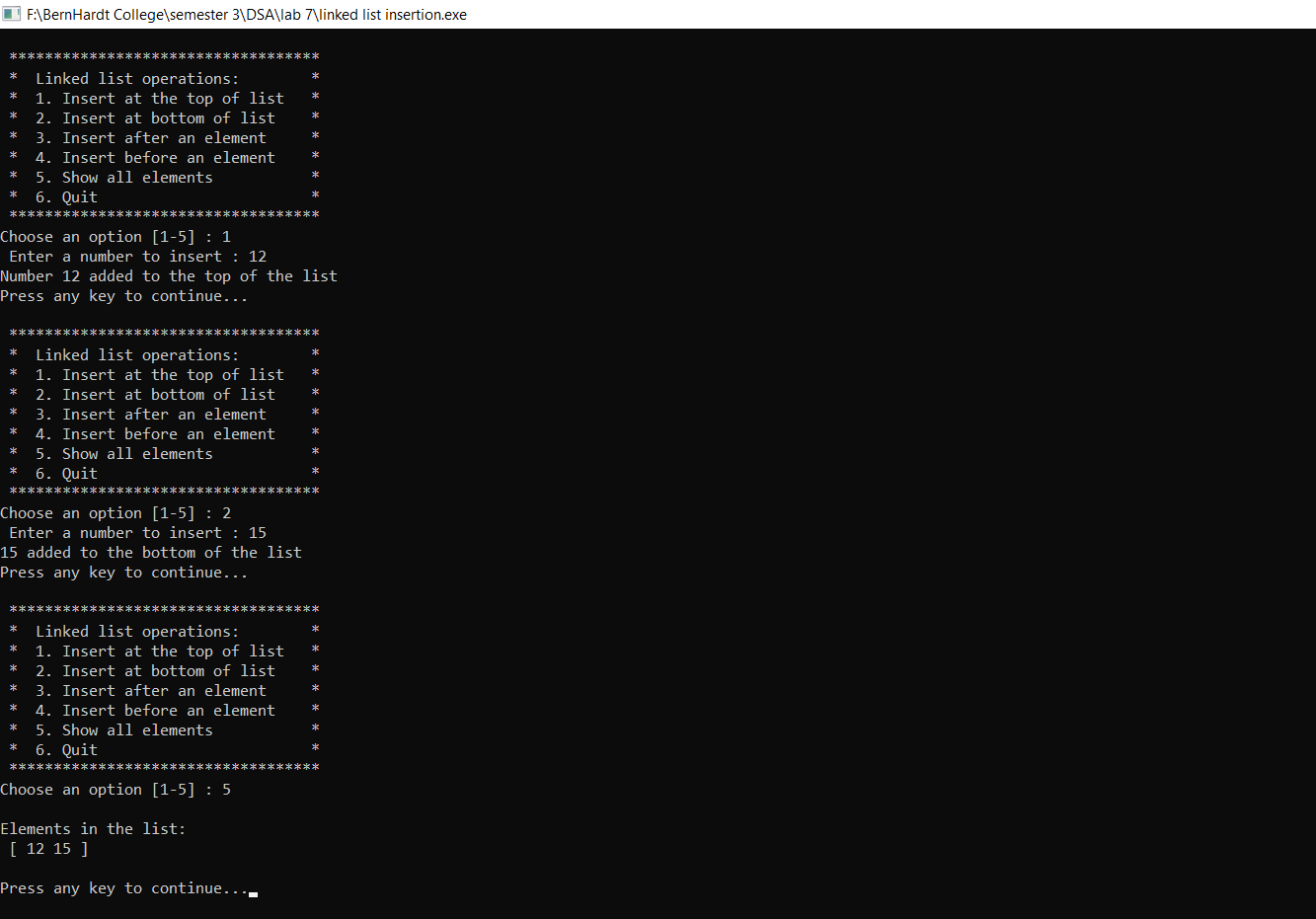
}printf("\n"); /\* End of Switch \*/

} /\* End of While \*/

return(0);

}

**Program Output**

****

**Write a program showing the concept of tree traversal.**

#include <stdio.h>

#include <stdlib.h>

/\* 1

/ \

2 3

/\ /\

4 5 6 7

\*/

/\* 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;

};

/\* 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);

}

/\* Given a binary tree, print its nodes according to the

"bottom-up" postorder traversal. \*/

void printPostorder(struct node\* node)

{

if (node == NULL)

return;

// first recur on left subtree

printPostorder(node->left);

// then recur on right subtree

printPostorder(node->right);

// now deal with the node

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

}

/\* Given a binary tree, print its nodes in inorder\*/

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);

}

/\* Given a binary tree, print its nodes in preorder\*/

void printPreorder(struct node\* node)

{

if (node == NULL)

return;

/\* first print data of node \*/

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

/\* then recur on left sutree \*/

printPreorder(node->left);

/\* now recur on right subtree \*/

printPreorder(node->right);

}

int main()

{

struct node \*root = newNode(1);

root->left = newNode(2);

root->right = newNode(3);

root->left->left = newNode(4);

root->left->right = newNode(5);

root->right->left= newNode(6);

root->right->right= newNode(7);

printf("Preorder traversal of binary tree is \n");

printPreorder(root);

printf("\nInorder traversal of binary tree is \n");

printInorder(root);

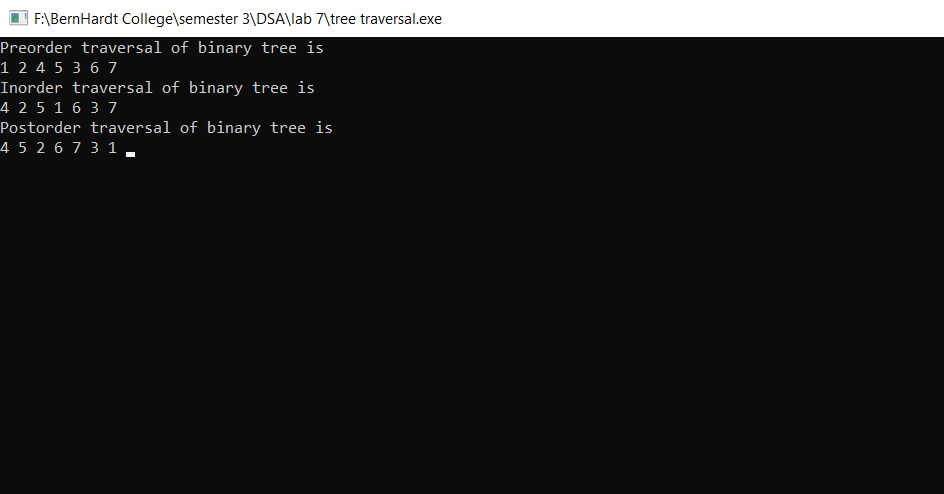
printf("\nPostorder traversal of binary tree is \n");

printPostorder(root);

getchar();

return 0;}

**Program Output**

****