**B.M.S. College of Engineering**

***(Autonomous Institution affiliated to VTU, Belagavi)***

**Department of Computer Science and Engineering**



**ACADEMIC YEAR: 2020-2021**

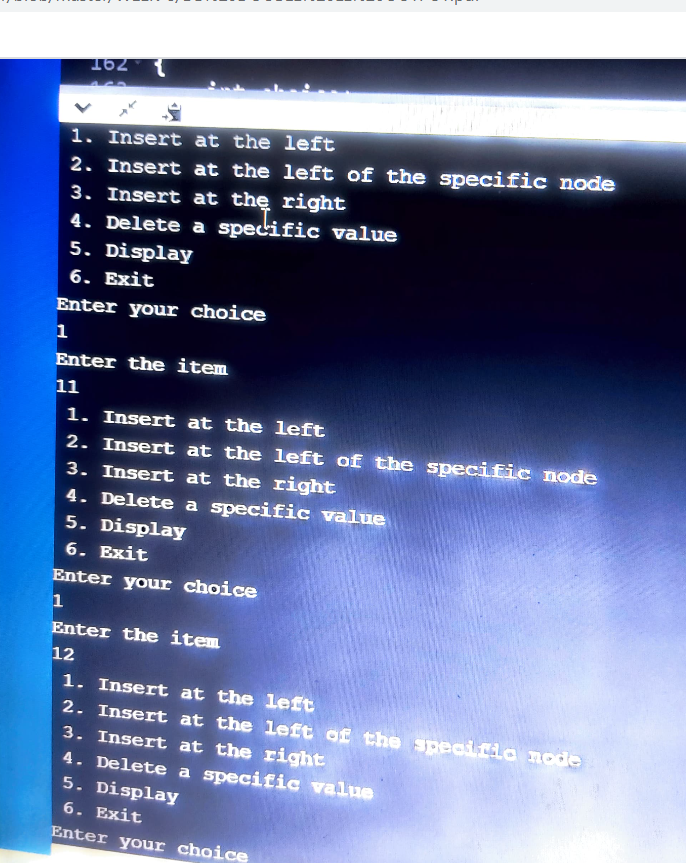
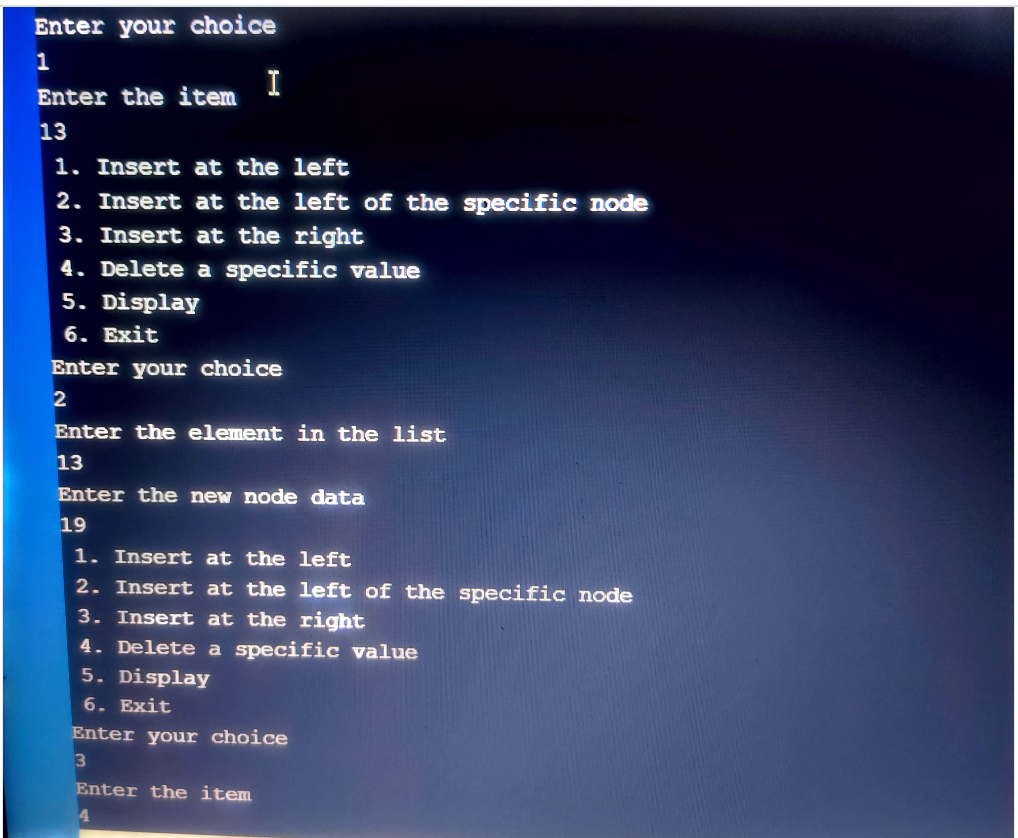
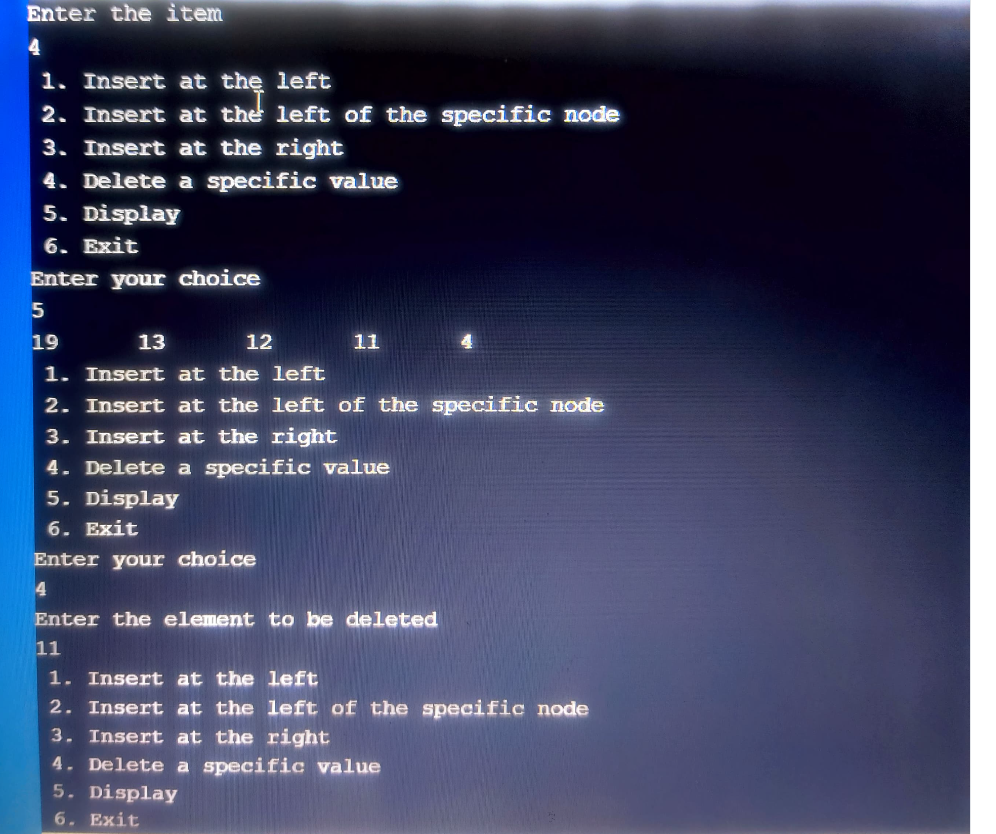
**NAME: AJITH MS**

**USN: 1BM19CS010**

**CLASS: CSE-3**

**SUBJECT: DATA STRUCTURES LAB RECORD**

|  |
| --- |
|  |
|  |
| **LAB-1** |
| **1.**   |  | | --- | |  | | #include <stdio.h> | | |  |  | | |  | int top = -1; | | |  | void push(int stack[], int ele); | | |  | int pop(int stack[]); | | |  | void display(int stack[]); | | |  |  | | |  | int main() | | |  | { | | |  | int stack[5]; | | |  | int i, choice, ele; | | |  |  | | |  | do | | |  | { | | |  | printf("---MENU---\n"); | | |  | printf("1. Push\n"); | | |  | printf("2. Pop\n"); | | |  | printf("3. Display\n"); | | |  | printf("4. Exit\n"); | | |  | printf("Enter your choice!\n"); | | |  | scanf("%d", &choice); | | |  | switch(choice) | | |  | { | | |  | case 1: | | |  | printf("Enter the element that you want to Push :\n"); | | |  | scanf("%d", &ele); | | |  | push(stack, ele); | | |  | break; | | |  |  | | |  | case 2: | | |  | ele = pop(stack); | | |  | if (ele == -1) | | |  | printf("Stack Underflow\n"); | | |  | else | | |  | printf("The Poped element is : %d\n", ele); | | |  | break; | | |  | case 3: display(stack); | | |  | break; | | |  | case 4: printf("EXITING....\n"); | | |  | break; | | |  | default: printf("Invalid choice!\n"); | | |  | } | | |  | } | | |  | while(choice != 4); | | |  | return 0; | | |  | } | | |  |  | | |  | void push(int stack[], int ele) | | |  | { | | |  | if (top==4) | | |  | { | | |  | printf("Stack overflow"); | | |  | } | | |  | else | | |  | { | | |  | top++; | | |  | stack[top]=ele; | | |  | } | | |  | } | | |  |  | | |  | int pop(int stack[]) | | |  | { | | |  | int popele; | | |  | if(top==-1) | | |  |  | | |  | return -1; | | |  |  | | |  | else | | |  | { | | |  | popele=stack[top]; | | |  | top--; | | |  | return (popele); | | |  | } | | |  | } | | |  |  | | |  | void display(int stack[]) | | |  | { | | |  | int i; | | |  | printf("The stack elements\n"); | | |  | for(i=top;i>=0;i--) | | |  | { | | |  |  | | |  | printf("%d\n",stack[i]); | | |  | } | | |  | }  C:\Users\Ajith\Pictures\Screenshots\Screenshot (89).png | |   C:\Users\Ajith\Pictures\Screenshots\Screenshot (90).png  **LAB-2** |
| **2**.   |  | | --- | | #include <stdio.h> | |  | # define MAX 100 | |  | char stack[MAX]; | |  | int top=-1; | |  |  | |  | void push(char ch) | |  | { | |  | if (top==MAX-1) | |  | printf("Stack is full\n"); | |  | else | |  | { | |  | top++; | |  | stack[top]=ch; | |  | } | |  | } | |  | char pop() | |  | { | |  | char item; | |  | if (top==-1) | |  | printf("\n stack is empty !"); | |  | else | |  | { | |  | item=stack[top]; | |  | top--; | |  | return item; | |  | } | |  |  | |  | } | |  |  | |  | int stackempty() | |  | { | |  | if(top==-1) return 1; | |  | else return 0; | |  | } | |  |  | |  | char stacktop() | |  | { | |  | if( top==-1) | |  | printf("\n stack is empty!"); | |  | else | |  | return stack[top]; | |  | } | |  | int priority(char ch) | |  | { | |  | switch(ch) | |  | { | |  | case '+': | |  | case '-':return (1); | |  | case '\*': | |  | case '/':return (2); | |  | case '^': return (3); | |  | default : return (0); | |  | } | |  | } | |  |  | |  |  | |  |  | |  | int main(int argc, char \*\*argv) | |  | { | |  | char infix[100]; | |  | int i, item; | |  | printf("Enter the infix expression :"); | |  | scanf("%s",infix); | |  | printf("Expression : %s",infix); | |  | printf("\n Postfix: "); | |  | i=0; | |  | while (infix[i]!='\0') | |  | { | |  |  | |  |  | |  | switch (infix[i]) | |  | { | |  | case '(': push(infix[i]); | |  | break; | |  | case ')':while(( item=pop())!='(') | |  | printf("%c",item); | |  | break; | |  | case '+': | |  | case '-': | |  | case '\*': | |  | case '/': | |  | case '^': | |  | while(!stackempty() && priority(infix[i])<=priority(stacktop())) | |  | { | |  | item=pop(); | |  |  | |  | printf("%c", item); | |  | } | |  |  | |  | push(infix[i]); | |  | break; | |  | default : printf("%c", infix[i]); | |  | break; | |  |  | |  |  | |  | } | |  | i++; | |  | } | |  |  | |  | while(!stackempty()) | |  | { | |  | char item; | |  | item=pop(); | |  | printf("%c", item); | |  |  | |  | } | |  | printf("\n"); | |  | return 0; | |  |  | |  | }  C:\Users\Ajith\Pictures\Screenshots\Screenshot (91).png  **LAB-3** |   **3.**   |  | | --- | | #include<stdio.h> | |  |  | |  | #include<stdlib.h> | |  | #define maxsize 5 | |  | void enqueue(int \*Q,int \*front, int \*rear) | |  | { | |  | int ele; | |  | if(\*rear>=maxsize-1) | |  | { | |  | printf("Queue is full.\n"); | |  | return; | |  | } | |  | if(\*front==-1) | |  | { | |  | (\*front)++; | |  | } | |  | (\*rear)++; | |  | printf("Enter the element to be inserted"); | |  | scanf("%d",&ele); | |  | \*(Q+\*rear)=ele; | |  | } | |  |  | |  | void display(int \*Q,int \*front,int \*rear) | |  | { | |  | if(\*front==-1&&\*rear==-1) | |  | printf("Queue is empty!!!!.\n"); | |  | else | |  | { | |  | printf("Elements in Queue are:\n"); | |  | for(int i=\*front;i<=\*rear;i++) | |  | { | |  | printf("%d ",\*(Q+i)); | |  | } | |  | printf("\n"); | |  | } | |  | } | |  |  | |  | void dequeue(int \*Q,int \*front, int \*rear) | |  | { | |  | int ele; | |  | if(\*front==-1&&\*rear==-1) | |  | { | |  | printf("Queue is empty!!!!\n"); | |  | return; | |  | } | |  | else if(\*front==\*rear) | |  | { | |  | ele=\*(Q+\*front); | |  | \*front=-1; | |  | \*rear=-1; | |  | } | |  | else | |  | { | |  | ele=\*(Q+\*front); | |  | (\*front)++; | |  | } | |  | printf("Deleted Element are: %d\n",ele); | |  | } | |  |  | |  | void main() | |  | { | |  | int front1=-1,rear1=-1; | |  | int queue1[maxsize]; | |  | int choice; | |  | printf("1. Enqueue\n"); | |  | printf("2. Dequque\n"); | |  | printf("3. Display\n"); | |  | printf("4. Exit\n"); | |  |  | |  | do | |  | { | |  | printf("Enter your choice"); | |  | scanf("%d",&choice); | |  |  | |  | switch(choice) | |  | { | |  | case 1: enqueue(queue1,&front1,&rear1); | |  | break; | |  | case 2: dequeue(queue1,&front1,&rear1); | |  | break; | |  | case 3: display(queue1,&front1,&rear1); | |  | break; | |  | case 4:exit(0); | |  | break; | |  | default:printf("Please input correct choice\n"); | |  | break; | |  |  | |  | } | |  | }while(choice!=4); | |  |  | |  | }  C:\Users\Ajith\Pictures\Screenshots\Screenshot (135).png | | **LAB-4** |  |   **4**.   |  | | --- | | #include<stdio.h> | |  | #include<stdlib.h> | |  | #define QUE\_SIZE 5 | |  | int item, front=0, rear=-1, q[QUE\_SIZE],count=0; | |  | void insertrear() | |  | { | |  | if(count == QUE\_SIZE) | |  | { | |  | printf("QUEUE OVERFLOW \n"); | |  | return ; | |  | } | |  |  | |  | rear= (rear+1)%QUE\_SIZE; | |  | q[rear]= item; | |  | count++; | |  | } | |  |  | |  | int deletefront() | |  | { | |  | if(count==0) | |  | {return -1; | |  | } | |  | item=q[front]; | |  | front=(front+1)%QUE\_SIZE; | |  | count=count-1; | |  | return item; | |  | } | |  |  | |  | void display() | |  | { | |  | int i,f; | |  | if(count==0) | |  | { | |  | printf("QUEUE IS EMPTY \n"); | |  | return; | |  | } | |  | f=front; | |  |  | |  | printf("CONTENTS OF QUEUE \n"); | |  | for(i=1; i<=count; i++) | |  | { | |  | printf("%d \n", q[f]); | |  | f=(f+1)%QUE\_SIZE; | |  | } | |  | } | |  |  | |  | void main() | |  | { | |  | int choice ; | |  | for(;;) | |  | { | |  | printf("ENTER 1.insert rear 2.delete front 3.DISPLAY \n"); | |  | printf("ENTER CHOICE \n"); | |  | scanf("%d", &choice); | |  | switch(choice) | |  | { | |  | case 1: printf("ENTER ITEM TO BE ENTERED \n"); | |  | scanf("%d", &item); | |  | insertrear(); | |  | break; | |  | case 2: item= deletefront(); | |  | if(item == -1) | |  | printf("QUEUE EMPTY \n"); | |  | else | |  | printf("ITEM DELETED IS %d", item); | |  | break; | |  | case 3: display(); | |  | break; | |  | default: exit(0); | |  | } | |  | } | |  | }  C:\Users\Ajith\Pictures\Screenshots\Screenshot (99).png  **LAB-5** |   **5**.   |  | | --- | | #include <stdio.h> | |  | #include <stdlib.h> | |  | #include <string.h> | |  | struct node | |  | { | |  | int sem; | |  | char name[100]; | |  | char usn[100]; | |  | struct node \*next; | |  | }; | |  | struct node \*head = NULL; | |  | int counter = 0; | |  |  | |  |  | |  | void | |  | Insertst () | |  | { | |  | struct node \*newnode; | |  | int s; | |  | char n[100], u[100]; | |  | printf ("\t ----Enter the name----- : "); | |  | scanf ("%s", n); | |  | printf ("\t ----Enter the semester----- : "); | |  | scanf ("%d", &s); | |  | printf ("\t ----Enter the usn---- : "); | |  | scanf ("%s", u); | |  | newnode = (struct node \*) malloc (sizeof (struct node)); | |  | newnode->sem = s; | |  | strcpy (newnode->name, n); | |  | strcpy (newnode->usn, u); | |  | if (head == NULL) | |  | printf (">>First node created\n"); | |  | newnode->next = head; | |  | head = newnode; | |  | counter++; | |  | printf ("Node created!\n"); | |  | } | |  |  | |  | void | |  | Insertany (int p) | |  | { | |  | struct node \*newnode; | |  | int s; | |  | char n[100], u[100]; | |  | printf ("\t -Enter the name- : "); | |  | scanf ("%s", n); | |  | printf ("\t -Enter the semester- : "); | |  | scanf ("%d", &s); | |  | printf ("\t -Enter the usn- : "); | |  | scanf ("%s", u); | |  | newnode = (struct node \*) malloc (sizeof (struct node)); | |  | newnode->sem = s; | |  | strcpy (newnode->name, n); | |  | strcpy (newnode->usn, u); | |  | if (p == 1) | |  | { | |  | printf ("Node of linked list is inserted in the first position\n"); | |  | newnode->next = head; | |  | head = newnode; | |  | counter++; | |  | } | |  | else if (head == NULL && p > 1) | |  | { | |  | printf ("currently empty!!!\n"); | |  | return; | |  | } | |  | else if (p > (counter + 1)) | |  | { | |  | printf | |  | ("Not possible since number of pre-existing nodes in list is insufficient!\n"); | |  | return; | |  | } | |  | else | |  | { | |  | struct node \*temp1; | |  | struct node \*temp2; | |  | int count = 1; | |  | temp1 = head; | |  | while (count < (p - 1)) | |  | { | |  | temp1 = temp1->next; | |  | count++; | |  | } | |  | temp2 = temp1->next; | |  | temp1->next = newnode; | |  | newnode->next = temp2; | |  | counter++; | |  | printf ("Node inserted at %d position in linked list\n", p); | |  | } | |  | } | |  |  | |  | void | |  | Insertend () | |  | { | |  | struct node \*newnode; | |  | struct node \*temp; | |  | int s; | |  | char n[100], u[100]; | |  | printf ("Enter the name- : "); | |  | scanf ("%s", n); | |  | printf ("Enter the semester- : "); | |  | scanf ("%d", &s); | |  | printf ("Enter the usn- : "); | |  | scanf ("%s", u); | |  | newnode = (struct node \*) malloc (sizeof (struct node)); | |  | newnode->sem = s; | |  | strcpy (newnode->name, n); | |  | strcpy (newnode->usn, u); | |  | if (head == NULL) | |  | { | |  | newnode->next = NULL; | |  | head = newnode; | |  | printf (">Very first node created\n"); | |  | counter++; | |  | } | |  | else | |  | { | |  | temp = head; | |  | while (temp->next != NULL) | |  | { | |  | temp = temp->next; | |  | } | |  | temp->next = newnode; | |  | newnode->next = NULL; | |  | counter++; | |  | printf ("Node created!\n"); | |  | } | |  | } | |  |  | |  | void | |  | display () | |  | { | |  | struct node \*ptr; | |  | ptr = head; | |  | int i = 1; | |  |  | |  | if (ptr == NULL) | |  | { | |  | printf ("Linked list is empty!!\n"); | |  | } | |  | else | |  | { | |  | while (ptr != NULL) | |  | { | |  | printf ("NODE %d\n", i); | |  | printf ("Name: %s\n", ptr->name); | |  | printf ("USN: %s\n", ptr->usn); | |  | printf ("Sem: %d\n", ptr->sem); | |  | printf ("---------\n"); | |  | i++; | |  | ptr = ptr->next; | |  | } | |  |  | |  | } | |  | } | |  | int | |  | main () | |  | { | |  | int choice, pos; | |  | do | |  | { | |  | printf ("\n"); | |  | printf | |  | ("\n1. Insert node at starting.. \n2. Insert node anywhere... \n3. Insert at the end of list.\n4. < Display list >\n5. Exit!!!\n"); | |  | printf ("\n\t Enter your choice() : "); | |  | scanf ("%d", &choice); | |  | if (choice == 5) | |  | break; | |  | switch (choice) | |  | { | |  | case 1: | |  | Insertst (); | |  | break; | |  |  | |  | case 2: | |  | printf ("Enter in which position you want to enter your node\n"); | |  | scanf ("%d", &pos); | |  | Insertany (pos); | |  | break; | |  |  | |  | case 3: | |  | Insertend (); | |  | break; | |  |  | |  | case 4: | |  | display (); | |  | break; | |  |  | |  | default: | |  | printf ("invalid!\n"); | |  | break; | |  | } | |  | } | |  | while (choice != 5); | |  | return 0; | |  | }  C:\Users\Ajith\Pictures\Screenshots\Screenshot (123).png  **C:\Users\Ajith\Pictures\Screenshots\Screenshot (124).png**  **C:\Users\Ajith\Pictures\Screenshots\Screenshot (125).png**  **LAB-6** |   **6**.   |  | | --- | | #include <stdio.h> | |  | #include <stdlib.h> | |  | #include <string.h> | |  | void create(); | |  | void display(); | |  | void insertpos(int); | |  | void insert\_beg(); | |  | void delete(); | |  | void delpos(int); | |  | void del\_beg(); | |  | struct node | |  | { | |  | int sem,usn; | |  | char name[20]; | |  | struct node \*next; | |  | }; | |  |  | |  | struct node \*head=NULL; | |  | int count=0; | |  | int main(int argc, char \*\*argv) | |  | { | |  | int choice,ele,a; | |  |  | |  | do | |  | { | |  | printf("\n1.Insert at the end \n2.Insert at the beginning \n3. Insert at a position \n4.Delete at the end \n5.Delete at the beginning \n6.Delete at a position \n7.Display\n8.Exit"); | |  | printf("\nEnter your choice : "); | |  | scanf("%d",&choice); | |  | switch(choice) | |  | { | |  | case 1: create(); break; | |  | case 2:insert\_beg(); | |  | break; | |  | case 3: printf("Enter the position to be inserted\n"); | |  | scanf("%d",&ele); | |  | insertpos(ele); | |  | break; | |  | case 4:delete();break; | |  | case 5:del\_beg();break; | |  | case 6:printf("enter the position"); | |  | scanf("%d",&a); | |  | delpos(a); | |  |  | |  | break; | |  | case 7:display(); | |  | break; | |  |  | |  | case 8:exit(0); | |  | } | |  |  | |  | }while(choice!=8); | |  | } | |  |  | |  | void create() | |  | { | |  | struct node \*newnode,\*temp; | |  | int sem1,usn1; | |  | char name1[20]; | |  | printf("Enter the name USN semester of student : "); | |  | scanf("%s %d %d",name1,&usn1,&sem1); | |  | newnode =(struct node \*) malloc (sizeof(struct node)); | |  |  | |  | strcpy(newnode->name,name1); | |  | newnode->usn=usn1; | |  | newnode->sem=sem1; | |  | if (head==NULL) | |  | { | |  | newnode->next=NULL; | |  | head=newnode; | |  | printf("Node created\n"); | |  | } | |  | else | |  | { | |  | temp=head; | |  | while(temp->next!=NULL) | |  | { | |  | temp=temp->next; | |  | } | |  | temp->next=newnode; | |  | newnode->next=NULL; | |  | printf("Node created\n"); | |  | count++; | |  | } | |  | } | |  |  | |  | void display() | |  | { | |  | struct node \*ptr=NULL; | |  | ptr=head; | |  |  | |  | if(ptr==NULL) | |  | { | |  | printf("Nothing to print\n"); | |  | } | |  | else | |  | { | |  | while(ptr!=NULL) | |  | { | |  | printf("%s ",ptr->name); | |  | printf("%d ",ptr->usn); | |  | printf("%d ",ptr->sem); | |  | printf("\n"); | |  | ptr=ptr->next; | |  | } | |  | } | |  |  | |  | } | |  |  | |  | void insertpos(int p) | |  | { | |  | struct node \*newnode; | |  | int sem1,usn1; | |  | char name1[20]; | |  | if(count+2<p) | |  | printf("the position exceeds the number of nodes"); | |  | else if(head==NULL&& p>1) | |  | { | |  | printf("node empty enter in first position"); | |  | } | |  | else | |  | { | |  | printf("Enter the name USN semester of student : "); | |  | scanf("%s %d %d",name1,&usn1,&sem1); | |  | newnode =(struct node \*) malloc (sizeof(struct node)); | |  |  | |  | strcpy(newnode->name,name1); | |  | newnode->usn=usn1; | |  | newnode->sem=sem1; | |  |  | |  |  | |  |  | |  | if(p==1) | |  | { | |  | printf("inserted at the beginning\n"); | |  | newnode->next=head; | |  | head=newnode; | |  | count++; | |  |  | |  | } | |  |  | |  | else | |  | { | |  |  | |  |  | |  | int i; | |  | struct node \*temp1; | |  | temp1=head; | |  | for(i=2;i<p;i++) | |  | { | |  | temp1= temp1->next; | |  | } | |  | newnode->next=temp1->next; | |  | temp1->next=newnode; | |  |  | |  | printf("Node inserted at %d position in linked list\n",p); | |  | count++; | |  | } | |  | } | |  | } | |  |  | |  | void insert\_beg() | |  | { | |  | struct node \*newnode; | |  | int sem1,usn1; | |  | char name1[20]; | |  | printf("Enter the name USN semester of student : "); | |  | scanf("%s %d %d",name1,&usn1,&sem1); | |  | newnode =(struct node \*) malloc (sizeof(struct node)); | |  |  | |  | strcpy(newnode->name,name1); | |  | newnode->usn=usn1; | |  | newnode->sem=sem1; | |  | newnode->next=head; | |  | head=newnode; | |  | count++; | |  | } | |  |  | |  | void delete() | |  | { | |  | struct node \*temp=NULL; | |  | int sem1,usn1; | |  | char name1[20]; | |  | if(head==NULL) | |  | printf("linked list is empty"); | |  | else | |  | { | |  | temp=head; | |  | while(temp->next->next!=NULL) | |  | { | |  | temp=temp->next; | |  | } | |  | strcpy(name1,temp->next->name); | |  | sem1=temp->next->sem; | |  | usn1=temp->next->usn; | |  | printf("the student info deleted = %s %d %d",name1,usn1,sem1); | |  | temp->next=NULL; | |  | count--; | |  | } | |  | } | |  |  | |  |  | |  | void del\_beg() | |  | { | |  | struct node \*temp=NULL; | |  | int sem1,usn1; | |  | char name1[20]; | |  | if(head==NULL) | |  | printf("linked list is empty"); | |  | else | |  | { | |  | strcpy(name1,head->name); | |  | sem1=head->sem; | |  | usn1=head->usn; | |  | printf("the student info deleted = %s %d %d",name1,usn1,sem1); | |  | temp=head; | |  | head=temp->next; | |  | free(temp); | |  | count--; | |  | } | |  | } | |  | void delpos(int p) | |  | { | |  | struct node \*temp=NULL; | |  | int sem1,usn1; | |  | char name1[20]; | |  | if(head==NULL) | |  | printf("linked list is empty"); | |  | else if(count+1<p) | |  | printf("the position exceeds the number of nodes"); | |  | else if(p==1) | |  | { | |  | strcpy(name1,head->name); | |  | sem1=head->sem; | |  | usn1=head->usn; | |  | printf("the student info deleted = %s %d %d",name1,usn1,sem1); | |  | temp=head; | |  | head=temp->next; | |  | free(temp); | |  | count--; | |  | } | |  | else | |  | { | |  | int i; | |  | struct node \*temp,\*ptr; | |  | temp=head; | |  | for(i=2;i<p;i++) | |  | { | |  | temp= temp->next; | |  | } | |  |  | |  | strcpy(name1,temp->next->name); | |  | sem1=temp->next->sem; | |  | usn1=temp->next->usn; | |  | printf("the student info deleted = %s %d %d",name1,usn1,sem1); | |  | ptr=temp->next; | |  | temp->next=temp->next->next; | |  | free(ptr); | |  | count--; | |  | } | |  |  | |  | }  C:\Users\Ajith\Pictures\Screenshots\Screenshot (107).pngC:\Users\Ajith\Pictures\Screenshots\Screenshot (106).pngC:\Users\Ajith\Pictures\Screenshots\Screenshot (104).pngC:\Users\Ajith\Pictures\Screenshots\Screenshot (103).pngC:\Users\Ajith\Pictures\Screenshots\Screenshot (102).pngC:\Users\Ajith\Pictures\Screenshots\Screenshot (101).pngC:\Users\Ajith\Pictures\Screenshots\Screenshot (100).png  **LAB-7** |   **7**   |  | | --- | | #include <stdlib.h> | |  | #include <string.h> | |  | struct node | |  | { | |  | int sem; | |  | struct node \*next; | |  | }; | |  | struct node \*head= NULL; | |  | struct node \*head2= NULL; | |  | int c=0; | |  | void Insert() | |  | { | |  | struct node \*newnode; | |  | struct node \*temp; | |  | int s; | |  | printf("Enter integer : "); | |  | scanf("%d",&s); | |  | newnode=(struct node\*)malloc(sizeof(struct node)); | |  | newnode->sem =s; | |  | if (head==NULL) | |  | { | |  | newnode->next=NULL; | |  | head=newnode; | |  | printf("first node of linked list created\n"); | |  | c++; | |  | } | |  | else | |  | { | |  | temp=head; | |  | while(temp->next!=NULL) | |  | { | |  | temp=temp->next; | |  | } | |  | temp->next=newnode; | |  | newnode->next=NULL; | |  | c++; | |  | printf("Node created\n"); | |  | } | |  | } | |  | void Insert2() | |  | { | |  | struct node \*newnode; | |  | struct node \*temp; | |  | int s,y; | |  | printf("enter elements to create list 2\n"); | |  | do | |  | { | |  | printf("Enter integer : \n"); | |  | scanf("%d",&s); | |  | newnode=(struct node\*)malloc(sizeof(struct node)); | |  | newnode->sem =s; | |  | if (head2==NULL) | |  | { | |  | newnode->next=NULL; | |  | head2=newnode; | |  | printf("first node of linked list created\n"); | |  | c++; | |  | } | |  | else | |  | { | |  | temp=head2; | |  | while(temp->next!=NULL) | |  | { | |  | temp=temp->next; | |  | } | |  | temp->next=newnode; | |  | newnode->next=NULL; | |  | c++; | |  | printf("Node created\n"); | |  | } | |  | printf("do u want to continue adding:0 or 1\n"); | |  | scanf("%d",&y); | |  | }while(y!=0); | |  | } | |  |  | |  |  | |  | void bubbleSort() | |  | { | |  | int swapped, i; | |  | struct node \*ptr1; | |  | struct node \*lptr = NULL; | |  |  | |  |  | |  | if (head == NULL) | |  | return; | |  |  | |  | do | |  | { | |  | swapped = 0; | |  | ptr1 = head; | |  |  | |  | while (ptr1->next != lptr) | |  | { | |  | if (ptr1->sem > ptr1->next->sem) | |  | { | |  | int temp = ptr1->sem; | |  | ptr1->sem = ptr1->next->sem; | |  | ptr1->next->sem = temp; | |  | swapped = 1; | |  | } | |  | ptr1 = ptr1->next; | |  | } | |  | lptr = ptr1; | |  | } | |  | while (swapped); | |  | } | |  |  | |  | void reverse() | |  | { | |  | struct node\* prev = NULL; | |  | struct node\* current = head; | |  | struct node\* next = NULL; | |  | while (current != NULL) { | |  | next = current->next; | |  | current->next = prev; | |  | prev = current; | |  | current = next; | |  | } | |  | head= prev; | |  | } | |  |  | |  | void concat() | |  | { | |  | struct node \*ptr; | |  | if(head==NULL) | |  | { | |  | head=head2; | |  | } | |  | if(head2==NULL) | |  | { | |  | head2=head; | |  | } | |  | ptr=head; | |  | while(ptr->next!=NULL) | |  | ptr=ptr->next; | |  | ptr->next=head2; | |  | } | |  | void display1() | |  | { | |  | struct node \*ptr; | |  | ptr=head; | |  | int i=1; | |  |  | |  | if(ptr==NULL) | |  | { | |  | printf("Linked list is empty!\n"); | |  | } | |  | else | |  | { | |  | while(ptr!= NULL) | |  | { | |  | printf(" %d",ptr->sem); | |  | i++; | |  | ptr=ptr->next; | |  | } | |  |  | |  | } | |  |  | |  | } | |  | void display2() | |  | { | |  | struct node \*ptr; | |  | ptr=head2; | |  | int i=1; | |  |  | |  | if(ptr==NULL) | |  | { | |  | printf("Linked list is empty!\n"); | |  | } | |  | else | |  | { | |  | while(ptr!= NULL) | |  | { | |  |  | |  |  | |  | printf(" %d",ptr->sem); | |  | printf("\n"); | |  | i++; | |  | ptr=ptr->next; | |  | } | |  |  | |  | } | |  |  | |  | } | |  |  | |  | int main() | |  | { | |  | int choice,pos; | |  | do | |  | { | |  |  | |  | printf("\n1. Insert node \n2. sort node\n3. reverse node\n4.concat 2 lists \n5.exit\n"); | |  | printf("\nEnter your choice : "); | |  | scanf("%d",&choice); | |  | switch(choice) | |  | { | |  | case 1: | |  | Insert(); | |  | break; | |  |  | |  | case 2: | |  | bubbleSort(); | |  | display1(); | |  | break; | |  |  | |  | case 3: | |  | reverse(); | |  | display1(); | |  | break; | |  |  | |  | case 4: | |  | Insert2(); | |  | concat(); | |  | display1(); | |  | break; | |  |  | |  | case 5: | |  | break; | |  |  | |  | default: | |  | printf("Wrong choice!\n"); | |  | break; | |  | } | |  | }while(choice!=5); | |  | return 0; | |  | }  C:\Users\Ajith\Pictures\Screenshots\Screenshot (109).pngC:\Users\Ajith\Pictures\Screenshots\Screenshot (108).png  C:\Users\Ajith\Pictures\Screenshots\Screenshot (112).pngC:\Users\Ajith\Pictures\Screenshots\Screenshot (111).pngC:\Users\Ajith\Pictures\Screenshots\Screenshot (110).png  **LAB-8** |   **8.1**   |  | | --- | | #include <stdio.h> | |  | #include <stdlib.h> | |  | struct node | |  | { | |  | int data; | |  | struct node \*next; | |  | }; | |  | void insert(); | |  | void display(); | |  | void del(); | |  |  | |  | struct node \*rear=NULL, \*front =NULL; | |  |  | |  | int main() | |  | { | |  | printf("\n--QUEUE IMPLEMENTATION USING LL--\n"); | |  | int choice; | |  |  | |  | do | |  | { | |  |  | |  | printf("\n1. Create \n2. Display \n3. Delete \n4. Exit \n"); | |  | printf("\nEnter your choice : "); | |  | scanf("%d",&choice); | |  | switch(choice) | |  | { | |  | case 1: insert(); break; | |  | case 2: display();break; | |  | case 3: del(); break; | |  | case 4: exit(0); | |  |  | |  | } | |  | }while(choice!=4); | |  | } | |  |  | |  | void insert() | |  | { | |  | struct node \*newnode; | |  | newnode=(struct node \*) malloc(sizeof(struct node)); | |  | printf("Enter the element:\n"); | |  | scanf("%d",&newnode->data); | |  | newnode->next=NULL; | |  |  | |  | if(rear==NULL) | |  | { | |  | rear=newnode; | |  | front=newnode; | |  |  | |  | } | |  | else | |  | { | |  | rear->next=newnode; | |  | rear=newnode; | |  | } | |  | } | |  |  | |  | void del() | |  | { | |  | if(front==NULL) | |  | { | |  | printf("Queue is empty\n");return; | |  | } | |  |  | |  | else | |  | { | |  | printf("Deleted element is %d",front->data); | |  | if(front==rear) | |  | { | |  | printf("\nQueue is empty\n"); | |  | front=NULL; rear=NULL; | |  | } | |  | else | |  | front=front->next; | |  | } | |  | } | |  |  | |  | void display() | |  | { | |  | struct node \*temp; | |  | if(front ==NULL) | |  | { | |  | printf("Queue is empty"); | |  | return; | |  | } | |  | temp=front; | |  | while (temp !=NULL) | |  | { | |  | printf("%d ",temp->data); | |  | temp=temp->next; | |  | } | |  |  | |  | } |   **8.2**   |  | | --- | | #include <stdio.h> | |  | #include<stdlib.h> | |  |  | |  | void push(); | |  | void pop(); | |  | void display(); | |  | struct node | |  | { | |  | int data; | |  | struct node \*next; | |  | }; | |  | struct node \*top=NULL; | |  |  | |  | int main() | |  | { | |  | int choice; | |  | printf("\n--STACK IMPLEMENTATION USING LL--\n"); | |  | do | |  | { | |  | printf("\n1. Push \n2. Display \n3. Pop\n4. Exit\n"); | |  | printf("\nEnter your choice : "); | |  | scanf("%d",&choice); | |  | switch(choice) | |  | { | |  | case 1: push(); break; | |  | case 2: display();break; | |  | case 3: pop(); break; | |  | case 4:exit(0); | |  | } | |  |  | |  | }while(choice!=4); | |  | } | |  |  | |  |  | |  | void push() | |  | { | |  | int item; | |  | struct node \*newnode; | |  | printf("Enter the element\n"); | |  | scanf("%d",&item); | |  |  | |  | newnode=(struct node\*)malloc(sizeof(struct node)); | |  | newnode->data=item; | |  | newnode->next=NULL; | |  | if(top==NULL) | |  | top=newnode; | |  | else | |  | newnode->next=top; | |  | top=newnode; | |  | } | |  | void pop() | |  | { | |  | if(top==NULL) | |  | printf("stack is empty"); | |  | else | |  | { | |  |  | |  | printf("element popped is %d", top->data); | |  |  | |  | top=top->next; | |  |  | |  | } | |  |  | |  |  | |  | } | |  |  | |  | void display() | |  | { | |  | struct node \*temp; | |  | temp=top; | |  | if(top==NULL) | |  | printf("Stack is empty"); | |  | while(temp!=NULL) | |  | { | |  | printf("%d ",temp->data); | |  | temp=temp->next; | |  | } | |  |  | |  | }  C:\Users\Ajith\Pictures\Screenshots\Screenshot (119).pngC:\Users\Ajith\Pictures\Screenshots\Screenshot (118).pngC:\Users\Ajith\Pictures\Screenshots\Screenshot (117).pngC:\Users\Ajith\Pictures\Screenshots\Screenshot (116).pngC:\Users\Ajith\Pictures\Screenshots\Screenshot (115).pngC:\Users\Ajith\Pictures\Screenshots\Screenshot (114).pngC:\Users\Ajith\Pictures\Screenshots\Screenshot (113).png  **LAB-9** |   **9**   |  | | --- | | #include<stdio.h> | |  | #include<stdlib.h> | |  | struct node | |  | { | |  | int data; | |  | struct node \*next; | |  | struct node \*prev; | |  | }; | |  | struct node \*head=NULL; | |  |  | |  | void insert\_left() | |  | { | |  | struct node \*new\_node; | |  | new\_node=(struct node\*)malloc(sizeof(struct node)); | |  | printf("Enter the item\n"); | |  | scanf("%d",&new\_node->data); | |  | new\_node->next=NULL; | |  | new\_node->prev=NULL; | |  |  | |  | if(head==NULL) | |  | { | |  | head=new\_node; | |  | } | |  | else | |  | { | |  | new\_node->next=head; | |  | head->prev=new\_node; | |  | head=new\_node; | |  | } | |  |  | |  | } | |  | void insert\_right() | |  | { | |  | struct node \*new\_node,\*temp; | |  | new\_node=(struct node\*)malloc(sizeof(struct node)); | |  | printf("Enter the item\n"); | |  | scanf("%d",&new\_node->data); | |  | new\_node->next=NULL; | |  | new\_node->prev=NULL; | |  | if(head==NULL) | |  | { | |  | head=new\_node; | |  | } | |  | else | |  | { | |  | temp=head; | |  | while(temp->next!=NULL) | |  | temp=temp->next; | |  | temp->next=new\_node; | |  | new\_node->prev=temp; | |  |  | |  | } | |  |  | |  | } | |  | void insert\_leftpos() | |  | { | |  | if(head==NULL) | |  | { | |  | printf("Empty list\n"); return; | |  | } | |  | int ele; | |  | struct node \*new\_node,\*temp; | |  | printf("Enter the element in the list\n"); | |  | scanf("%d",&ele); | |  | new\_node=(struct node\*)malloc(sizeof(struct node)); | |  | printf("Enter the new node data\n"); | |  | scanf("%d",&new\_node->data); | |  | new\_node->next=NULL; | |  | new\_node->prev=NULL; | |  |  | |  | temp=head; | |  | if(temp->data==ele) | |  | { | |  | new\_node->next=head; | |  | head->prev=new\_node; | |  | head=new\_node; | |  | } | |  | else if(temp->next==NULL) | |  | { | |  | printf("Element is not in the list\n"); | |  | } | |  | else | |  | { | |  | while(temp->next->data!=ele) | |  | { | |  | temp=temp->next; | |  | if(temp==NULL) | |  | { | |  | printf("Element is not in the list\n"); | |  | return; | |  | } | |  | } | |  |  | |  | new\_node->next=temp->next; | |  | temp->next=new\_node; | |  | new\_node->prev=temp; | |  | new\_node->next->prev=new\_node; | |  | } | |  |  | |  |  | |  | } | |  | void delete() | |  | { | |  | struct node \*temp; | |  | int ele; | |  | if(head==NULL) | |  | { | |  | printf("Empty List \n"); | |  | return; | |  | } | |  | printf("Enter the element to be deleted\n"); | |  | scanf("%d",&ele); | |  | temp=head; | |  | while(temp->data!=ele) | |  | { | |  | temp=temp->next; | |  | if(temp==NULL) | |  | { | |  | printf("Element is not in the list\n"); | |  | return; | |  | } | |  | } | |  | if(temp==head) //first node | |  | { | |  | head=head->next; | |  | } | |  | else if(temp->next==NULL) //last node | |  | { | |  | temp=temp->prev; | |  | temp->next=NULL; | |  | } | |  |  | |  | else //middle | |  | { | |  | temp->prev->next=temp->next; | |  | temp->next->prev=temp->prev; | |  | free(temp); | |  | } | |  |  | |  | } | |  | void display() | |  | { | |  | if(head==NULL) | |  | { | |  | printf("Empty List \n"); | |  | } | |  | else | |  | { | |  | struct node \*temp; | |  | temp=head; | |  | while(temp!=NULL) | |  | { | |  | printf("%d\t",temp->data); | |  | temp=temp->next; | |  | } | |  | printf("\n"); | |  | } | |  | } | |  |  | |  |  | |  | int main() | |  | { | |  | int choice; | |  | do | |  | { | |  | printf(" 1. Insert at the left \n"); | |  | printf(" 2. Insert at the left of the specific node \n"); | |  | printf(" 3. Insert at the right \n"); | |  | printf(" 4. Delete a specific value\n"); | |  | printf(" 5. Display\n"); | |  | printf(" 6. Exit\n"); | |  | printf("Enter your choice\n"); | |  | scanf("%d",&choice); | |  | switch(choice) | |  | { | |  | case 1: insert\_left(); break; | |  | case 2: insert\_leftpos(); break; | |  | case 3: insert\_right(); break; | |  | case 4: delete(); break; | |  | case 5: display(); break; | |  | case 6: exit(0); | |  | } | |  | }while(choice!=6); | |  | } | |

**LAB-10**

**10.**

#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node\* left;

struct node\* right;

}\*root1;

struct node \*create()

{

struct node \*temp;

printf("Enter ROOT NODE ELEMENT : ");

temp=(struct node\*)malloc(sizeof(struct node));

scanf("%d",&temp->data);

temp->left=temp->right=NULL;

return temp;

}

void insert(struct node \*root,struct node \*temp)

{

if(temp->data<root->data)

{

if(root->left!=NULL)

insert(root->left,temp);

else

root->left=temp;

}

if(temp->data>root->data)

{

if(root->right!=NULL)

insert(root->right,temp);

else

root->right=temp;

}

}

void printPostorder(struct node\* node)

{

if (node == NULL)

return;

printPostorder(node->left);

printPostorder(node->right);

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

}

void printInorder(struct node\* node)

{

if (node == NULL)

return;

printInorder(node->left);

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

printInorder(node->right);

}

void printPreorder(struct node\* node)

{

if (node == NULL)

return;

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

printPreorder(node->left);

printPreorder(node->right);

}

int main()

{

int choice;

struct node\* temp;

do

{

printf("\n-----MENU-----\n");

printf("1. CREATE\n");

printf("2. INSERT\n");

printf("3. PREORDER TRAVERSAL\n");

printf("4. INORDER TRAVERSAL\n");

printf("5. POSTORDER TRAVERSAL\n");

printf("6. EXIT\n");

printf("Enter your choice correctly : \n");

scanf("%d", &choice);

switch(choice)

{

case 1: root1 = create();

break;

case 2: printf("Enter the VALUE you want to INSERT : ");

temp=(struct node\*)malloc(sizeof(struct node));

scanf("%d",&temp->data);

insert(root1, temp);

break;

case 3: printPreorder(root1);

break;

case 4: printInorder(root1);

break;

case 5: printPostorder(root1);

break;

case 6: printf("EXITING...!!!");

break;

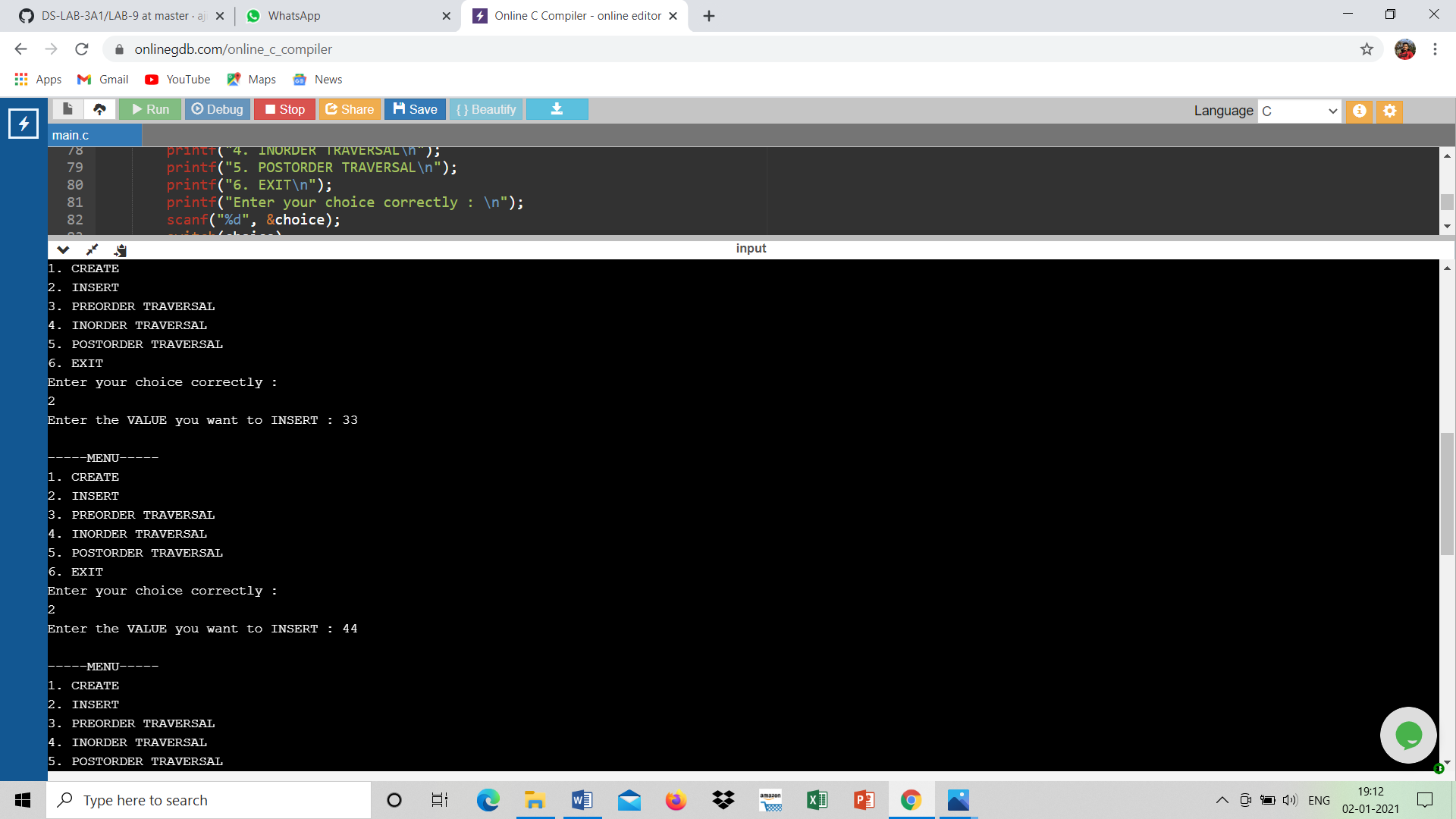
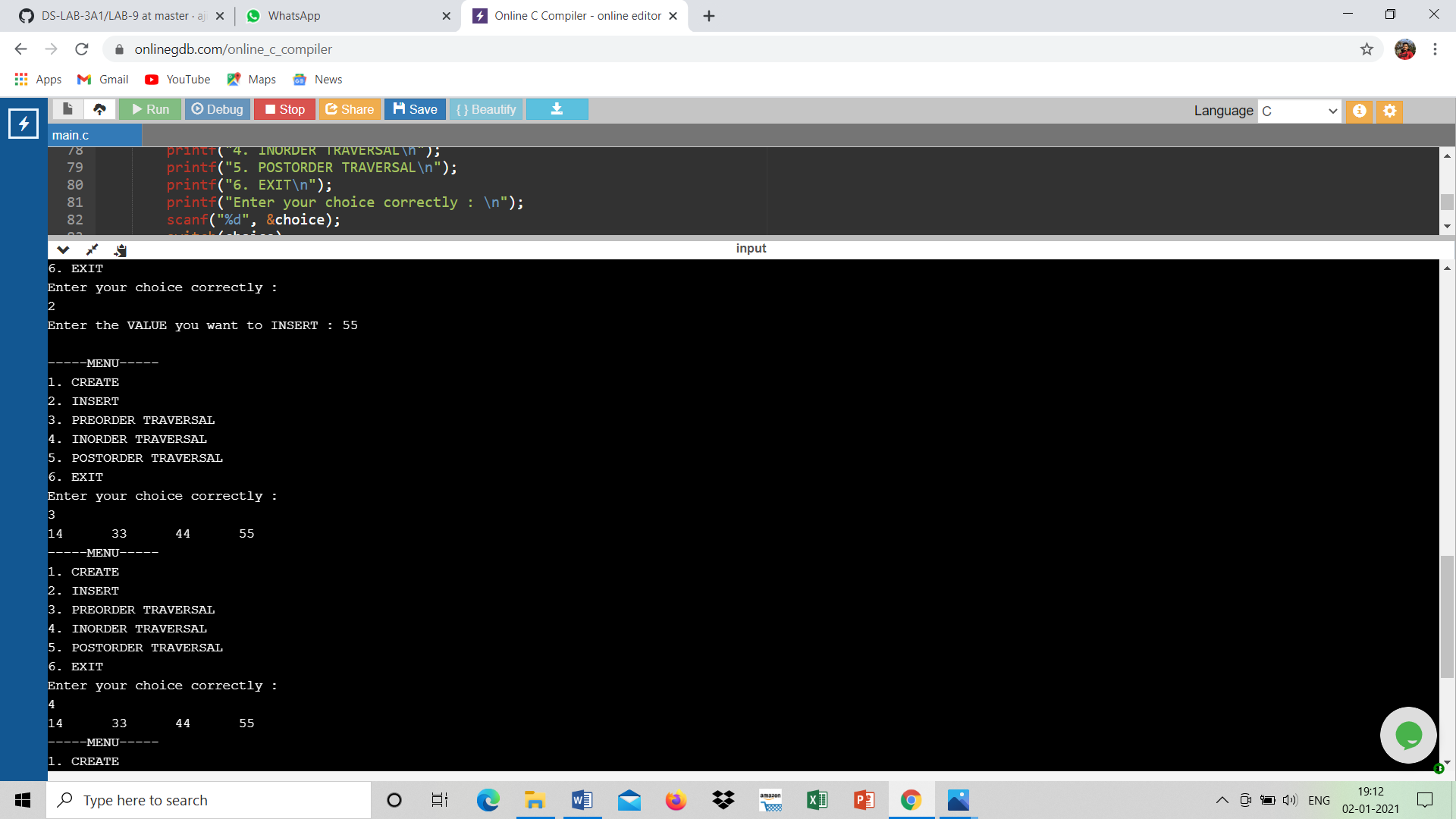
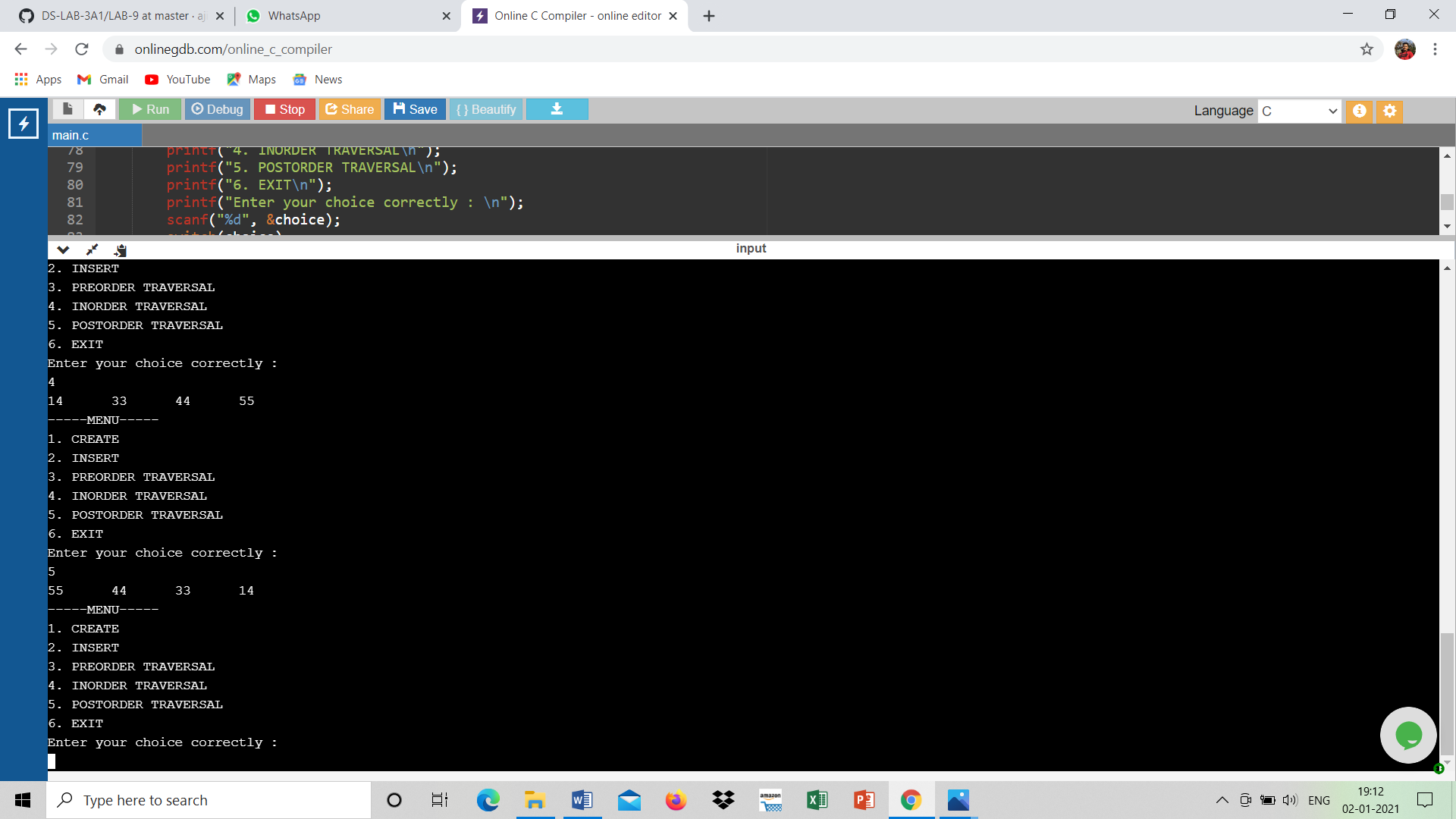
default: printf("Incorrect Choice!!\n");

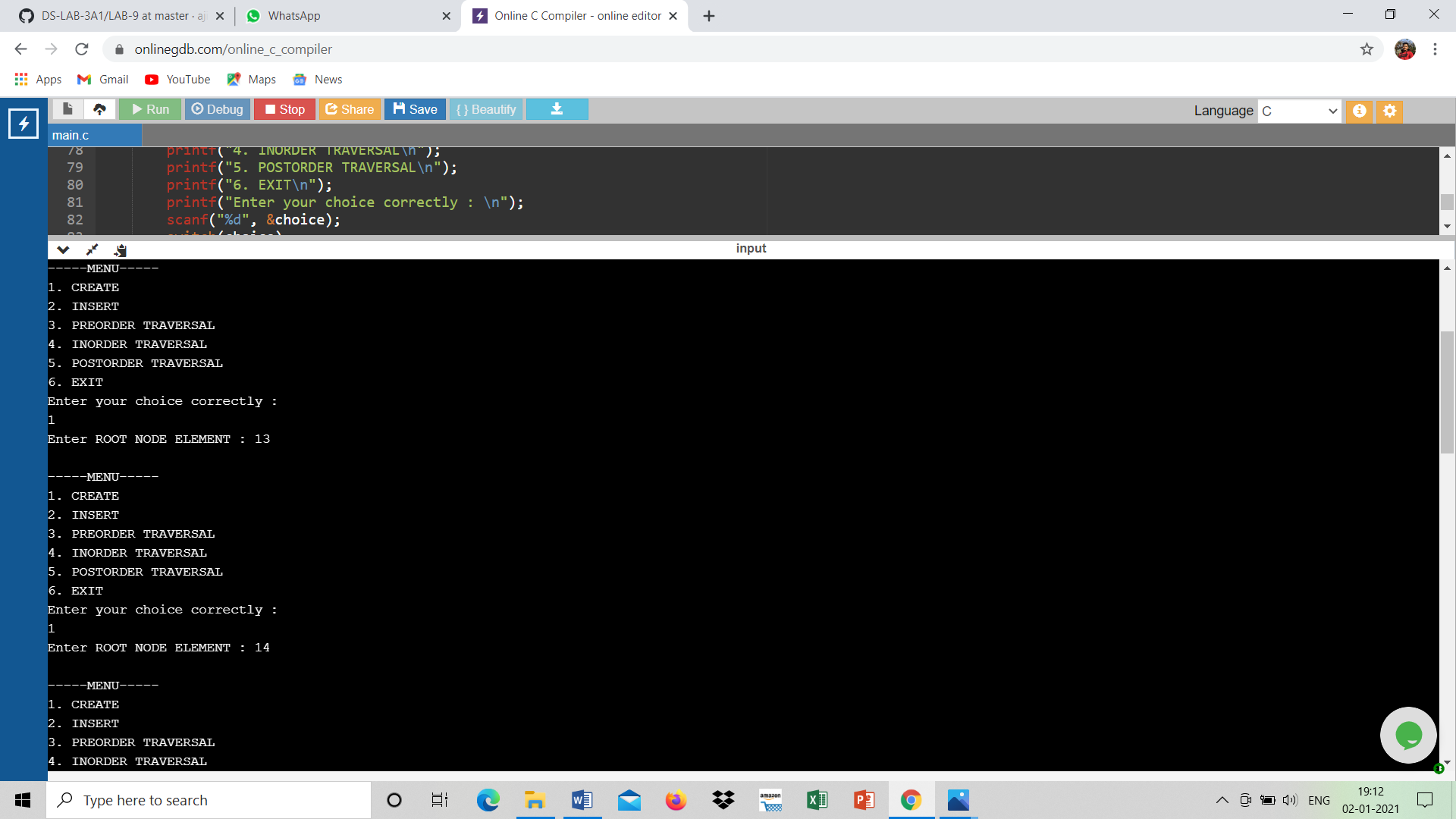
}

}while(choice != 6);

return 0;

}





**END…**

**THANK YOU**

**AJITH MS**

**1BM19CS010**

**CSE 3-A**