**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**“JnanaSangama”, Belgaum -590014, Karnataka.**



# DATA STRUCTURE LAB RECORD

***Submitted by***

**PRIYANSHU GUPTA**

**USN-1BM19CS124**

### *Under the Guidance of*

**Prof. SHEETAL VA**

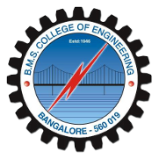
**Assistant Professor, BMSCE**

***in partial fulfillment for the award of the degree of***

**BACHELOR OF ENGINEERING**

***in***

## COMPUTER SCIENCE AND ENGINEERING



## B.M.S. COLLEGE OF ENGINEERING

**(Autonomous Institution under VTU)**

**BENGALURU-560019**

**Sep-2020 to Jan-2021**

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

**Bull Temple Road, Bangalore 560019**

(Affiliated To Visvesvaraya Technological University, Belgaum)

### Department of Computer Science and Engineering



#### CERTIFICATE

This is to certify that the LAB RECORD carried out by **PRIYANSHU GUPTA**

**(1BM19CS124)** who​ is the bonafide students of **B.**​ **M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor**​ **of Engineering in Computer Science and Engineering** of the Visveswaraiah Technological University, Belgaum during the year 2020-2021. The lab report has been approved as it satisfies the academic requirements in respect of **DATA**​ **STRUCTURE LAB RECORD (19CS3PCDST)** work​ prescribed for the said degree.

Signature of the Guide Signature of the HOD

Prof. Prof. Sheelal VA​ Dr. Umadevi ​Assistant Professor ​ Associate Prof.& Head, Dept. of CSE​

BMSCE, Bengaluru BMSCE, Bengaluru

External Viva

Name of the Examiner Signature with date

1.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**>>>Lab Program 1**

**Write a program to simulate the working of stack using an array with the following : (a) Push (b) Pop (c) Display. The program should print appropriate messages for stack overflow, stack underflow.**

#include <stdio.h>

#include<conio.h> #define STACK\_SIZE 3

int top=-1; int s[10]; int item; void push(){ if(top==STACK\_SIZE-1){

printf("Stack overflow\n"); return;

}

top=top+1; s[top]=item; } int pop(){ if(top==-1){ printf("stack is empty\n"); }else return s[top--]; } void display(){

int i;

if(top==-1){ printf("stack is empty\n"); return;

} printf("items in the stack are: "); for(i=top;i>=0;i--){ printf("%d\t",s[i]);

}

} main(){

int choice; int item\_deleted; for(;;){ printf("\n1:push 2:pop 3:display 4:exit\n"); printf("Enter your choice: "); scanf("%d",&choice); switch(choice){ case 1: printf("Enter the item to be inserted:"); scanf("%d",&item); push(); break;

case 2: item\_deleted=pop(); if(item\_deleted==-1) printf("stack is empty\n"); else

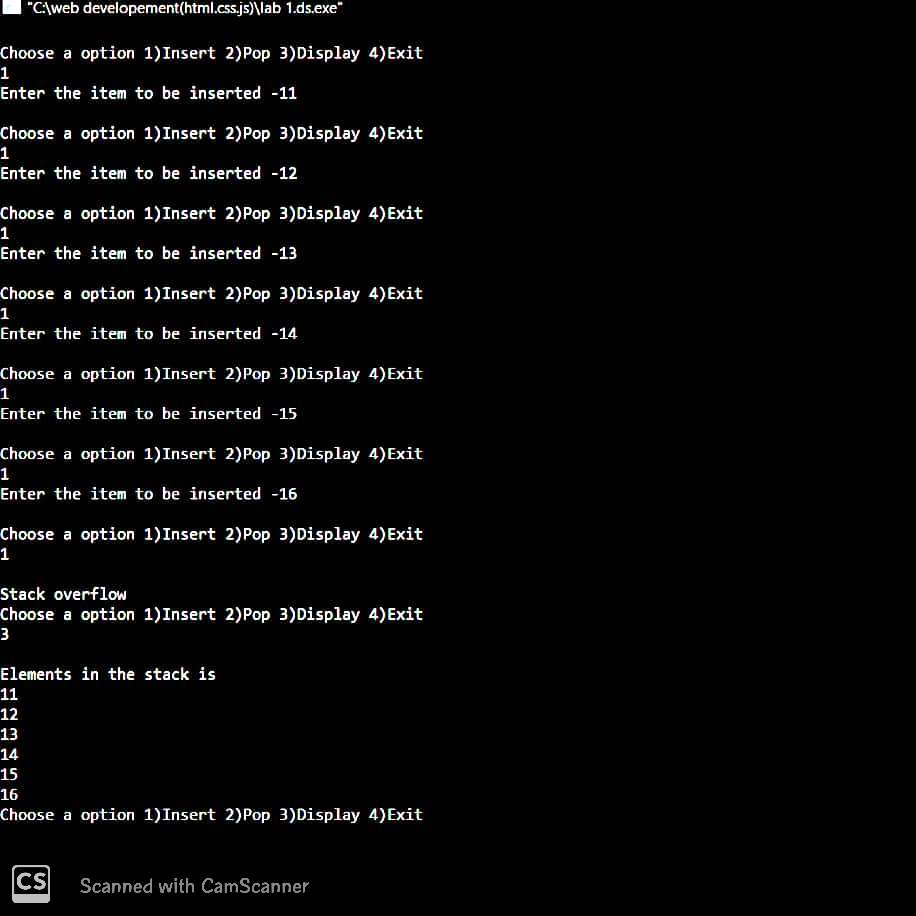
printf("item deleted is %d\n",item\_deleted); break; case 3: display(); break; default: exit(0);

}

}

}

**Output of lab 1:**



**==========================================================================**

**>>>Lab Program 2**

**WAP to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and the binary operators + (plus), - (minus), \* (multiply) and / (divide)**

#include<stdio.h> #include<string.h> int F (char symbol)

{ switch(symbol){

case '-':

case '+':return 2; case '\*': case '/':return 4; case '^': case '$': return 5; case '(': return 0; case '#': return -1; default: return 8;

} } int G (char symbol) { switch (symbol){

case '-': case '+':return 1; case '\*': case '/':return 3; case '^': case '$': return 6; case '(': return 9; case ')':return 0; case '#': return 0; default: return 7;

}

} void infix\_postfix(char infix[],char postfix[])

{ int top,i,j;

char s[30],symbol; top=-1; s[++top]='#'; j=0; for (i=0;i<strlen(infix);i++)

{ symbol=infix[i]; while (F(s[top])>G(symbol))

{ postfix [j]=s[top--]; j++; } if (F(s[top])!=G(symbol)) s[++top]=symbol;

else top--; } while (s[top]!='#')

{ postfix[j++]=s[top--];

} postfix[j]='\0';

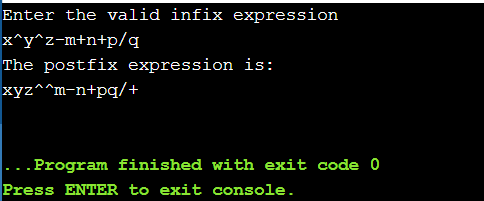
} int main()

{

char infix [20]; char postfix [20]; printf ("enter the valid infix expression\n"); scanf("%s",infix); infix\_postfix(infix,postfix); printf("the postfix expression is\n"); printf("%s\n",postfix);

}

**Output of lab 2:**



**==========================================================================**

**>>>Lab Program 3**

**WAP to simulate the working of a queue of integers using an array.Provide the following operations**

**a) Insert (b) Delete (c) Display**

**The program should print appropriate messages for queue empty and queue overflow conditions**

#include<stdio.h>

#include<stdlib.h>

#define QUE\_SIZE 3

int item,front=0,rear=-1,q[10];

void insertrear()

{

if(rear==QUE\_SIZE-1)

{ printf("queue overflow\n"); return; } rear=rear+1; q[rear]=item; } int deletefront(){ if (front>rear){ front=0; rear=-1; return -1;

}return q[front++]; } void displayQ(){

int i; if (front>rear)

{

printf("queue is empty\n"); return; } printf("contents of queue\n"); for(i=front;i<=rear;i++)

{ printf("%d\n",q[i]);

}} int main()

{ int choice; for(;;)

{ printf("1:insertrear 2:deletefront 3:display 4:exit"); printf("enter the choice\n"); scanf("%d",&choice); switch(choice){ case 1:printf("enter the item to be inserted\n"); scanf("%d",&item);

insertrear (); break; case 2:item=deletefront(); if(item==-1) printf("queue is empty\n");

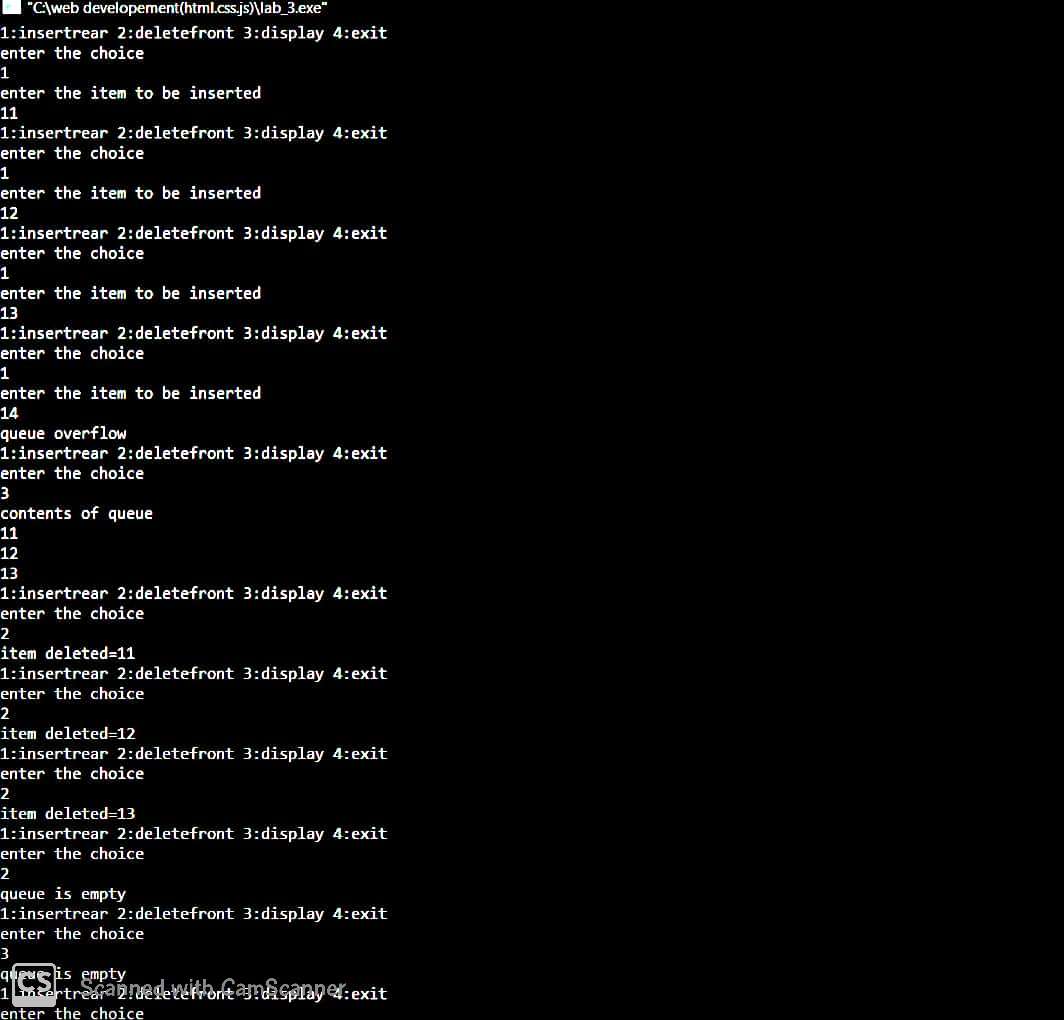
else printf("item deleted=%d\n",item); break; case 3:displayQ(); break;

default:exit (0);

} }

}

**Output of lab 3:**



**==========================================================================**

**>>>Lab Program 4**

**WAP to simulate the working of a circular queue of integers using an array. Provide the following operations.**

**a) Insert (b) Delete (c) Display**

**The program should print appropriate messages for queue empty and queue overflow conditions.**

#include<stdio.h>

#include<stdlib.h>

#define SIZE 3

int item; int front=0,rear=-1,q[SIZE],count=0; void insertrear(){ if(count==SIZE) { printf("\nqueue overflow\n"); return; } rear=(rear+1)%SIZE; q[rear]=item; count++; } int deletefront(){ if(count==0) return -1; item=q[front]; front=(front+1)%SIZE; count=count-1; return item; } void displayQ(){

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)%SIZE;

} } int main(){ int choice; for(;;){ printf("1.insertrear 2.deletefront 3.display 4.exit\n"); printf("enter choice\n"); scanf("%d",&choice); switch(choice) { case 1:printf("Enter the item to be inserted:\n "); scanf("%d",&item);

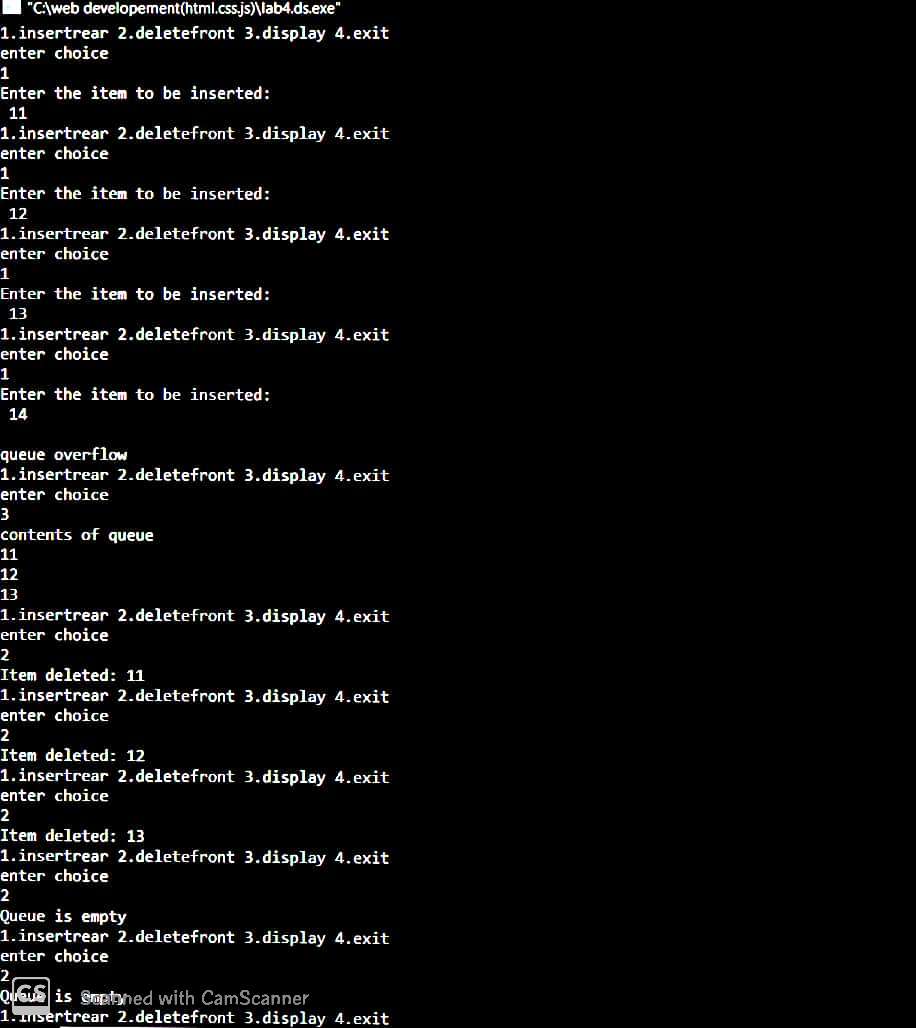
insertrear(); break; case 2:item=deletefront(); if(item==-1) printf("Queue is empty\n"); else printf("Item deleted: %d\n",item); break; case 3: displayQ(); break; default: exit(0);

}

}

}

**Output of lab 4:**



**==========================================================================**

**>>>Lab Program 5 and 6**

**WAP to Implement Singly Linked List with following operations**

1. **Create a linked list.**
2. **Insertion of a node at first position, at any position and at end of list**
3. **Deletion of first element, specified element and last element in the list.**
4. **Display the contents of the linked list.**

#include<stdio.h>

#include<conio.h> #include<process.h> struct node{ int info; struct node \*link;

};

typedef struct node \*NODE;

NODE getnode(){ NODE x; x=(NODE)malloc(sizeof(struct node)); if(x==NULL) { printf("memory full\n"); exit(0); } return x;

} void freenode(NODE x){

free(x);

}

NODE insert\_front(NODE first,int item){ NODE temp; temp=getnode();

temp->info=item; temp->link=NULL; if(first==NULL) return temp; temp->link=first; first=temp; return first;

}

NODE delete\_front(NODE first){ NODE temp; if(first==NULL) { printf("list is empty cannot delete\n"); return first; } temp=first; temp=temp->link; printf("item deleted at front-end is=%d\n",first->info); free(first); return temp;

}

NODE insert\_rear(NODE first,int item){ NODE temp,cur; temp=getnode();

temp->info=item; temp->link=NULL; if(first==NULL) return temp; cur=first; while(cur->link!=NULL) cur=cur->link; cur->link=temp;

return first;

}

NODE delete\_rear(NODE first){ NODE cur,prev; if(first==NULL) { printf("list is empty cannot delete\n"); return first; } if(first->link==NULL) { printf("item deleted is %d\n",first->info); free(first);

return NULL;

}

prev=NULL;

cur=first; while(cur->link!=NULL)

{ prev=cur; cur=cur->link; } printf("iten deleted at rear-end is %d",cur->info); free(cur);

prev->link=NULL;

return first;

}

NODE insert\_pos(int item,int pos,NODE first){

NODE temp; NODE prev,cur;

int count; temp=getnode(); temp->info=item; temp->link=NULL; if(first==NULL && pos==1) return temp; if(first==NULL){ printf("invalid pos\n"); return first;

}

if(pos==1){ temp->link=first; return temp; } count=1; prev=NULL;

cur=first;

while(cur!=NULL && count!=pos){ prev=cur; cur=cur->link; count++; } if(count==pos){ prev->link=temp; temp->link=cur;

return first; } printf("Invalid position\n"); return first;

}

NODE delete\_pos(int pos,NODE first){

NODE cur; NODE prev;

int count,flag=0;

if(first==NULL || pos<0){ printf("invalid position\n"); return NULL;

} if(pos==1){ cur=first; first=first->link; freenode(cur); return first;

}

prev=NULL;

cur=first; count=1; while(cur!=NULL){ if(count==pos){flag=1;break;} count++; prev=cur; cur=cur->link; } if(flag==0){ printf("invalid position\n"); return first; } printf("item deleted at given position is %d\n",cur->info); prev->link=cur->link; freenode(cur); return first; } void display(NODE first){ NODE temp; if(first==NULL) printf("list empty cannot display items\n"); for(temp=first;temp!=NULL;temp=temp->link) { printf("%d\n",temp->info);

} } void main(){ int item,choice,pos; NODE first=NULL;

for(;;){

printf("\n 1:Insert\_front 2:Delete\_front 3:Insert\_rear 4:Delete\_rear 5:insert\_pos 6:delete\_pos

7:display\_list 8:Exit\n"); printf("enter the choice="); scanf("%d",&choice); switch(choice) { case 1:printf("enter the item at front-end="); scanf("%d",&item); first=insert\_front(first,item);

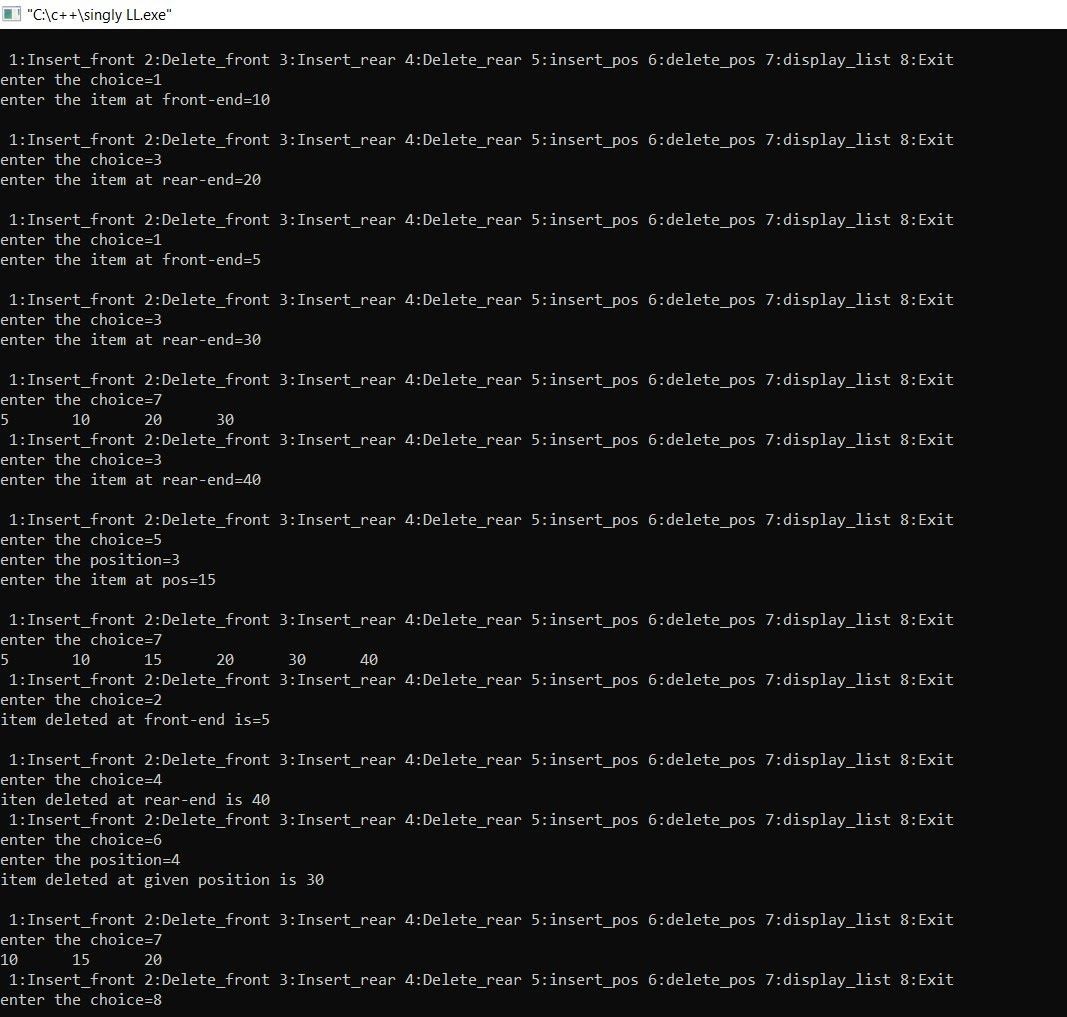
break; case 2:first=delete\_front(first); break; case 3:printf("enter the item at rear-end="); scanf("%d",&item); first=insert\_rear(first,item); break; case 4:first=delete\_rear(first); break; case 5:printf("enter the position="); scanf("%d",&pos); printf("enter the item at pos="); scanf("%d",&item); first=insert\_pos(item,pos,first); break; case 6:printf("enter the position="); scanf("%d",&pos); first=delete\_pos(pos,first); break; case 7:display(first); break; default:exit(0); break;

}

}

}

**Output of lab 5 and 6:**



**==========================================================================**

**>>>Lab Program 7**

**WAP Implement Single Link List with following operations**

1. **Sort the linked list.**
2. **Reverse the linked list.**
3. **Concatenation of two linked lists**

#include<stdio.h>

#include<conio.h> #include<stdlib.h> struct node{ int info; struct node \*link;

};

typedef struct node \*NODE;

NODE getnode(){

NODE x; x=(NODE)malloc(sizeof(struct node)); if(x==NULL) {

printf("mem full\n"); exit(0); } return x;

}

NODE insert\_rear(NODE first,int item){ NODE temp,cur; temp=getnode(); temp->info=item; temp->link=NULL; if(first==NULL) return temp; cur=first; while(cur->link!=NULL) cur=cur->link; cur->link=temp;

return first; } void display(NODE first){ NODE temp; if(first==NULL) printf("list empty");

for(temp=first;temp!=NULL;temp=temp->link){ printf("%d ",temp->info);

} printf("\n");

}

NODE concat(NODE first,NODE second){ NODE cur; if(first==NULL) return second; if(second==NULL)

return first; cur=first;

while(cur->link!=NULL) cur=cur->link; cur->link=second;

return first;

}

NODE reverse(NODE first){ NODE cur,temp; cur=NULL; while(first!=NULL) { temp=first; first=first->link; temp->link=cur; cur=temp;

} return cur; } void sort(NODE first){ int swapped, i; NODE ptr1; NODE lptr=NULL;

if (first == NULL)

return; do

{ swapped = 0; ptr1 = first; while (ptr1->link != lptr) { if (ptr1->info > ptr1->link->info){ swap(ptr1, ptr1->link); swapped = 1;

} ptr1 = ptr1->link;

} lptr = ptr1; }

while (swapped);

}

/\* function to swap data of two nodes a and b\*/ void swap( NODE a, NODE b){

int temp = a->info; a->info = b->info; b->info = temp;

} void main(){ int item,choice,pos,i,n; NODE a,b;

for(;;){ printf("1.concatenate 2.sort 3.reverse 4.dislay 5.exit\n"); printf("enter the choice="); scanf("%d",&choice); switch(choice) { case 1:printf("Enter the no of nodes in 1 :"); scanf("%d",&n); a=NULL;

printf("Enter the items :\n"); for(i=0;i<n;i++) { scanf("%d",&item); a=insert\_rear(a,item);

} printf("Enter the no of nodes in 2 :"); scanf("%d",&n); b=NULL;

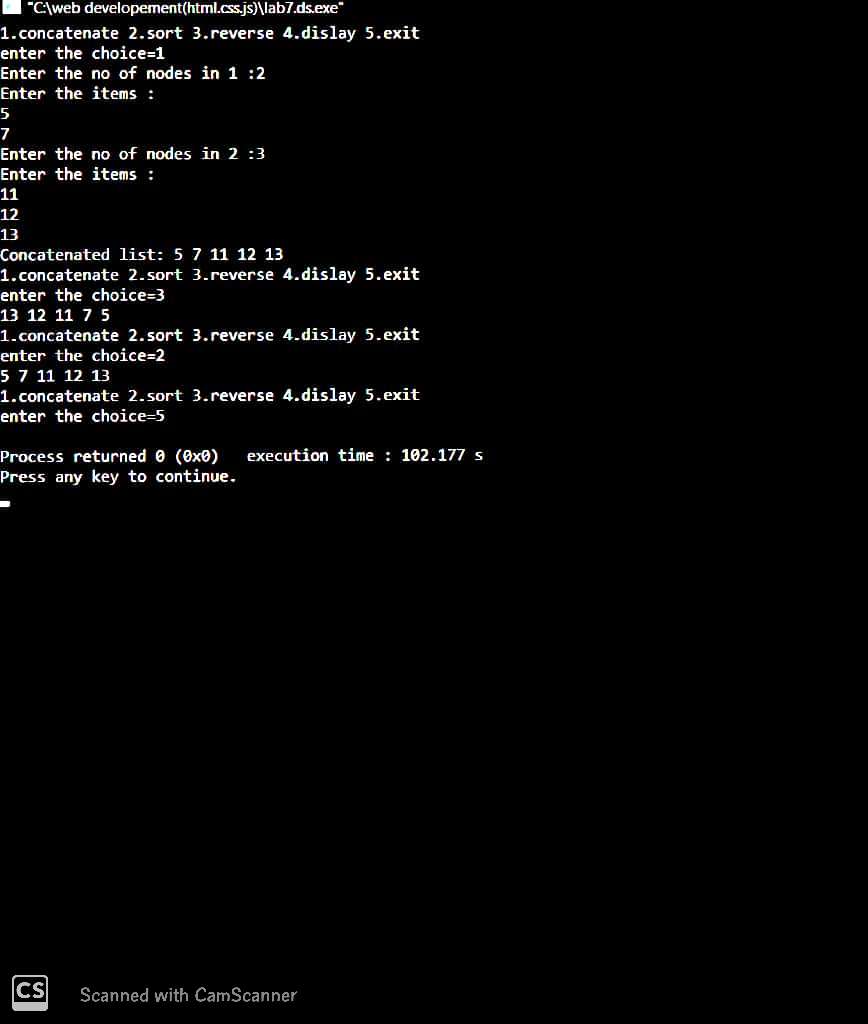
printf("Enter the items :\n"); for(i=0;i<n;i++) { scanf("%d",&item); b=insert\_rear(b,item);

} a=concat(a,b); printf("Concatenated list: "); case 2:sort(a); display(a); break; case 3:a=reverse(a); display(a); break; case 4:display(a);

break; default:exit(0);

} } getch(); }

**Output of lab7:**



**==========================================================================**

**>>>Lab Program 8**

**WAP to implement Queue using Linked Representation**

#include<stdio.h>

#include<conio.h> #include<process.h> struct node { int info; struct node \*link;

};

typedef struct node \*NODE;

NODE getnode()

{

NODE x; x=(NODE)malloc(sizeof(struct node)); if(x==NULL)

{ printf("mem full\n"); exit(0); } return x;

} void freenode(NODE x)

{ free(x);

}

NODE insert\_rear(NODE first,int item)

{

NODE temp,cur; temp=getnode(); temp->info=item; temp->link=NULL; if(first==NULL) return temp; cur=first; while(cur->link!=NULL) cur=cur->link; cur->link=temp;

return first; }

NODE delete\_front(NODE first)

{

NODE temp; if(first==NULL) { printf("list is empty cannot delete\n"); return first; } temp=first; temp=temp->link; printf("item deleted at front-end is=%d\n",first->info); free(first); return temp; } void display(NODE first)

{

NODE temp; if(first==NULL) printf("list empty cannot display items\n"); for(temp=first;temp!=NULL;temp=temp->link)

{

printf("%d\n",temp->info);

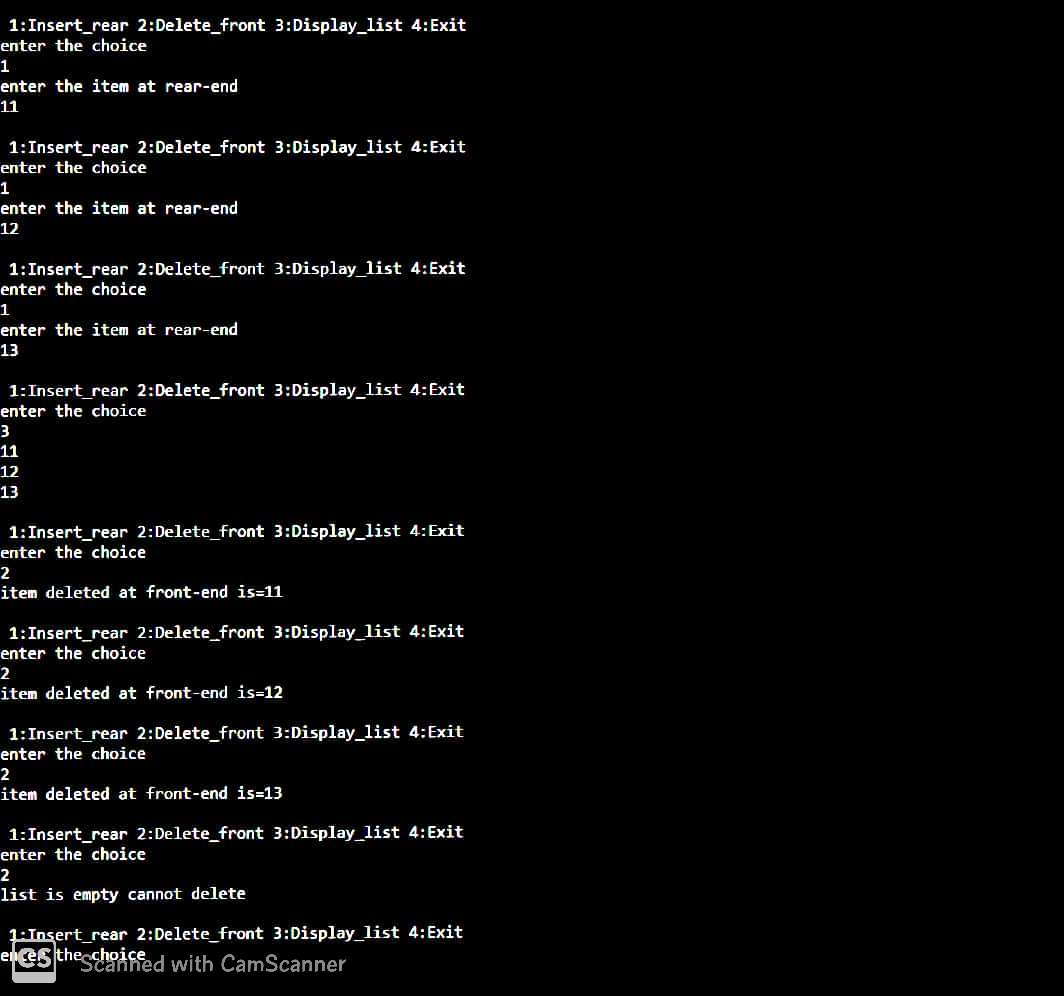
} } void main() { int item,choice,pos; NODE first=NULL;

for(;;) { printf("\n 1:Insert\_rear 2:Delete\_front 3:Display\_list 4:Exit\n"); printf("enter the choice\n"); scanf("%d",&choice); switch(choice)

{ case 1:printf("enter the item at rear-end\n"); scanf("%d",&item); first=insert\_rear(first,item); break; case 2:first=delete\_front(first); break; case 3:display(first); break; default:exit(0); break;

} } getch(); }

**Outputlab 8:**



**WAP to implement Stack using Linked Representation**

#include<stdio.h>

#include<conio.h> #include<process.h> struct node { int info; struct node \*link;

};

typedef struct node \*NODE;

NODE getnode()

{

NODE x; x=(NODE)malloc(sizeof(struct node)); if(x==NULL)

{ printf("mem full\n"); exit(0); } return x;

} void freenode(NODE x)

{ free(x);

}

NODE insert\_front(NODE first,int item)

{

NODE temp; temp=getnode(); temp->info=item; temp->link=NULL; if(first==NULL) return temp; temp->link=first; first=temp; return first;

}

NODE delete\_front(NODE first)

{

NODE temp; if(first==NULL) { printf("stack is empty cannot delete\n"); return first; } temp=first; temp=temp->link; printf("item deleted at front-end is=%d\n",first->info); free(first); return temp; } void display(NODE first)

{

NODE temp; if(first==NULL) printf("stack empty cannot display items\n"); for(temp=first;temp!=NULL;temp=temp->link)

{

printf("%d\t",temp->info);

} } void main() { int item,choice,pos;

NODE first=NULL;

for(;;) { printf("\n1:Insert\_front 2:Delete\_front 3:Display\_list 4:Exit\n"); printf("enter the choice="); scanf("%d",&choice); switch(choice)

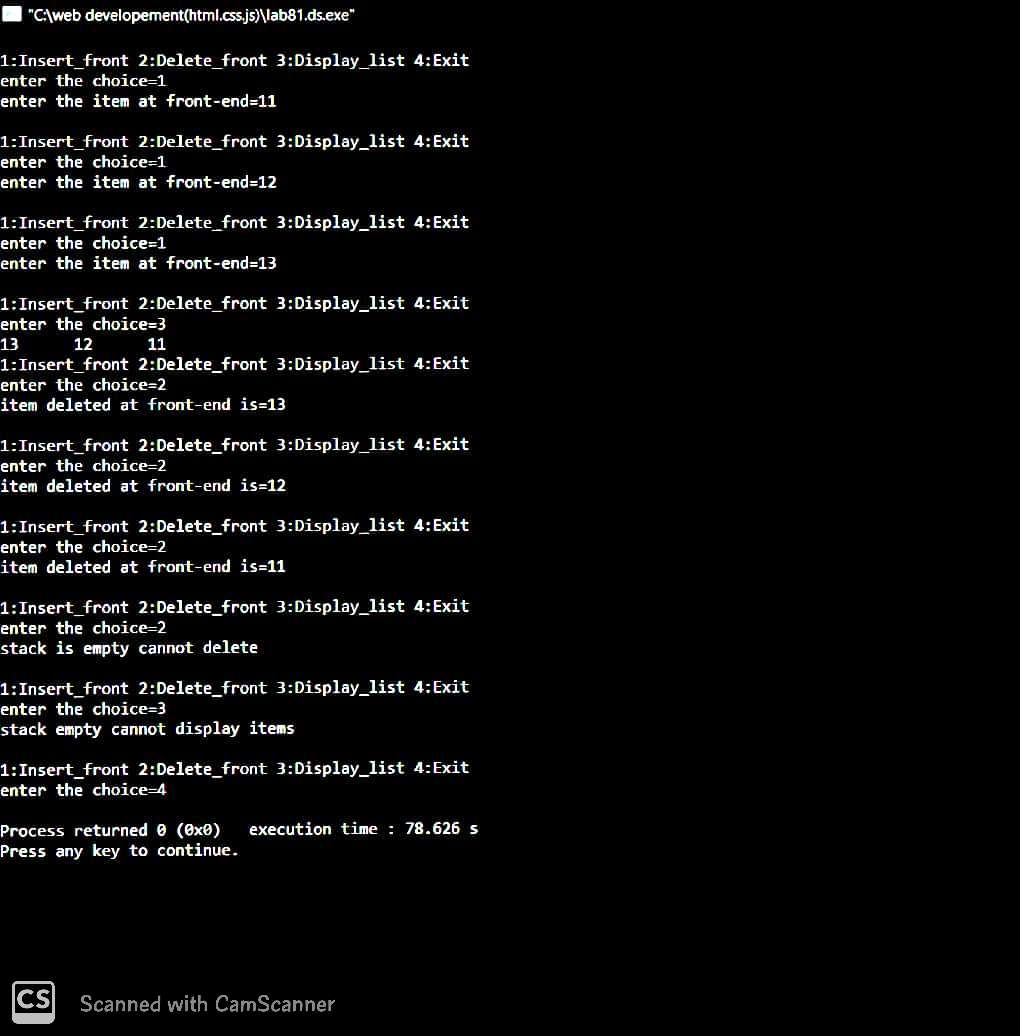
{ case 1:printf("enter the item at front-end="); scanf("%d",&item); first=insert\_front(first,item); break; case 2:first=delete\_front(first); break; case 3:display(first); break; default:exit(0); break;

}

}

}

**Output:**



**==========================================================================**

**>>>Lab Program 9**

**WAP Implement doubly link list with primitive operations**

1. **Create a doubly linked list.**
2. **Insert a new node to the left of the node.**
3. **Delete the node based on a specific value**
4. **Display the contents of the list**

#include<stdio.h> #include<stdlib.h> struct node{ int info; struct node \*llink; struct node \*rlink;

};

typedef struct node \*NODE;

NODE getnode(){ NODE x; x=(NODE)malloc(sizeof(struct node)); if(x==NULL){

printf("mem full\n");

exit(0);

} return x;

} void freenode(NODE x){

free(x);

}

NODE dinsert\_front(int item,NODE head){ NODE temp,cur; temp=getnode(); temp->info=item; cur=head->rlink; head->rlink=temp; temp->llink=head; temp->rlink=cur; cur->llink=temp; return head;

}

NODE dinsert\_rear(int item,NODE head){ NODE temp,cur; temp=getnode(); temp->info=item; cur=head->llink; head->llink=temp;

temp->rlink=head; temp->llink=cur; cur->rlink=temp; return head;

}

NODE ddelete\_front(NODE head){ NODE cur,next; if(head->rlink==head){ printf("dq empty\n"); return head; } cur=head->rlink; next=cur->rlink; head->rlink=next; next->llink=head; printf("the node deleted is %d",cur->info); freenode(cur); return head;

}

NODE ddelete\_rear(NODE head){ NODE cur,prev; if(head->rlink==head){ printf("dq empty\n"); return head;

} cur=head->llink; prev=cur->llink; head->llink=prev; prev->rlink=head; printf("the node deleted is %d",cur->info); freenode(cur); return head;

}

NODE delete\_all\_key(int item,NODE head){ NODE prev,cur,next; int count; if(head->rlink==head) { printf("LE"); return head;

}

count=0; cur=head->rlink; while(cur!=head){ if(item!=cur->info) cur=cur->rlink; else { count++; prev=cur->llink; next=cur->rlink; prev->rlink=next; next->llink=prev; freenode(cur); cur=next;

} } if(count==0) printf("key not found"); else printf("key found at %d positions and are deleted", count); return head;

} void display(NODE head){ NODE temp; if(head->rlink==head){ printf("dq empty"); return; } printf("contents of dq="); temp=head->rlink; while(temp!=head){ printf("%d ",temp->info); temp=temp->rlink; } printf("\n");

}

NODE delete\_except\_one(int item,NODE head){ NODE prev,cur,next; int count; if(head->rlink==head) { printf("LE"); return head;

}

count=0; cur=head->rlink; while(cur!=head){ if(item!=cur->info) cur=cur->rlink; else { count++; if(count==1){ cur=cur->rlink;} else { prev=cur->llink; next=cur->rlink; prev->rlink=next; next->llink=prev; freenode(cur); cur=next;

}

} } if(count==0) printf("key not found"); else printf("key found at %d positions and are deleted.", count); return head;

}

NODE insert\_leftpos(int item,NODE head){ NODE temp,cur,prev; if(head->rlink==head){ printf("list empty"); return head; } cur=head->rlink; while(cur!=head){ if(item==cur->info)break; cur=cur->rlink; } prev=cur->llink;

printf("enter towards left of %d=",item); temp=getnode(); scanf("%d",&temp->info); prev->rlink=temp; temp->llink=prev; cur->llink=temp; temp->rlink=cur; return head;

}

NODE insert\_rightpos(int item,NODE head){ NODE temp,cur,next; if(head->rlink==head){ printf("list empty"); return head; } cur=head->rlink; while(cur!=head){ if(item==cur->info)break; cur=cur->rlink; } next=cur->rlink; printf("enter towards right of %d=",item); temp=getnode(); scanf("%d",&temp->info);

next->llink=temp; cur->rlink=temp; temp->llink=cur; temp->rlink=next; return head;

} int main(){ NODE head,last; int item, choice; head=getnode(); head->rlink=head; head->llink=head;

for(;;){

printf("\n1:insert\_front 2:insert\_rear 3:delete\_front 4:delete\_rear 5:display 6:delete\_duplicates

7:delete\_all\_except\_one 8:insert\_left 9:insert\_right 10:exit\n"); printf("enter the choice="); scanf("%d",&choice); switch(choice){ case 1: printf("enter the item at front end="); scanf("%d",&item);

last=dinsert\_front(item,head); break; case 2: printf("enter the item at rear end="); scanf("%d",&item); last=dinsert\_rear(item,head); break; case 3:last=ddelete\_front(head); break; case 4: last=ddelete\_rear(head); break; case 5: display(head); break; case 6: printf("Enter the item whose duplicates are to be deleted: "); scanf("%d",&item); head=delete\_all\_key(item,head);

break;

case 7: printf("Enter the item to be deleted: "); scanf("%d",&item); head=delete\_except\_one(item,head);

break;

case 8: printf("enter the key item="); scanf("%d",&item); head=insert\_leftpos(item,head); break;

case 9: printf("Enter key item="); scanf("%d",&item); head=insert\_rightpos(item,head);

break;

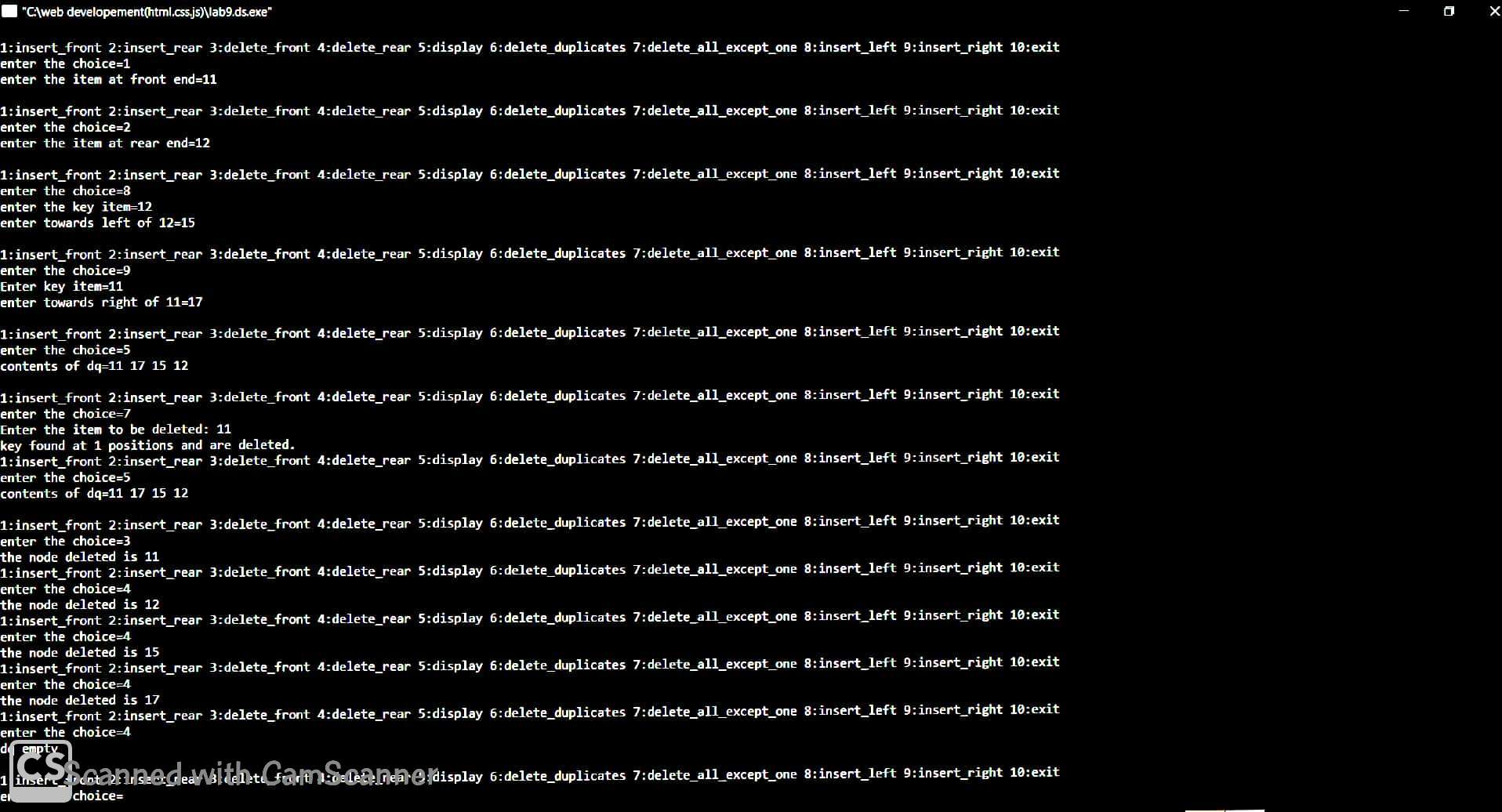
default:exit(0);

}

}

}

**Output:**



**==========================================================================**

**>>LabProgram10 Write a program a) To construct a**

**binary Search tree.**

#include<stdio.h>

#include<conio.h> #include<process.h> struct node{ int info; struct node \*rlink; struct node \*llink;

};

typedef struct node \*NODE; NODE getnode(){

NODE x; x=(NODE)malloc(sizeof(struct node)); if(x==NULL){

printf("mem full\n"); exit(0); } return x;

} void freenode(NODE x){

free(x);

}

NODE insert(NODE root,int item){ NODE temp,cur,prev; temp=getnode(); temp->rlink=NULL; temp->llink=NULL; temp->info=item; if(root==NULL) return temp;

prev=NULL;

cur=root; while(cur!=NULL){ prev=cur; cur=(item<cur->info)?cur->llink:cur->rlink; } if(item<prev->info) prev->llink=temp; else prev->rlink=temp; return root; } void display(NODE root,int i){

int j;

if(root!=NULL){ display(root->rlink,i+1); for(j=0;j<i;j++) printf(" "); printf("%d\n",root->info); display(root->llink,i+1);

}

}

NODE delete(NODE root,int item){ NODE cur,parent,q,suc; if(root==NULL){ printf("empty\n"); return root;

}

parent=NULL;

cur=root; while(cur!=NULL&&item!=cur->info){ parent=cur; cur=(item<cur->info)?cur->llink:cur->rlink;

}

if(cur==NULL){

printf("not found\n"); return root; } if(cur->llink==NULL) q=cur->rlink; else if(cur->rlink==NULL) q=cur->llink; else { suc=cur->rlink; while(suc->llink!=NULL) suc=suc->llink; suc->llink=cur->llink; q=cur->rlink;

}

if(parent==NULL)

return q; if(cur==parent->llink) parent->llink=q;

else parent->rlink=q; freenode(cur); return root;

}

void preorder(NODE root){ if(root!=NULL){ printf("%d ",root->info); preorder(root->llink); preorder(root->rlink);

}

}

void postorder(NODE root){ if(root!=NULL){ postorder(root->llink); postorder(root->rlink); printf("%d ",root->info);

}

}

void inorder(NODE root){ if(root!=NULL){ inorder(root->llink); printf("%d ",root->info); inorder(root->rlink);

}

}

void main(){ int item,choice;

NODE root=NULL;

for(;;){ printf("\n1.insert 2.preorder 3.postorder 4.inorder 5.delete 6.display 7.exit\n"); printf("enter the choice="); scanf("%d",&choice); switch(choice)

{ case 1:printf("enter the item="); scanf("%d",&item); root=insert(root,item); break; case 2:preorder(root); break; case 3:postorder(root); break; case 4:inorder(root); break; case 5:printf("enter the item="); scanf("%d",&item); root=delete(root,item);

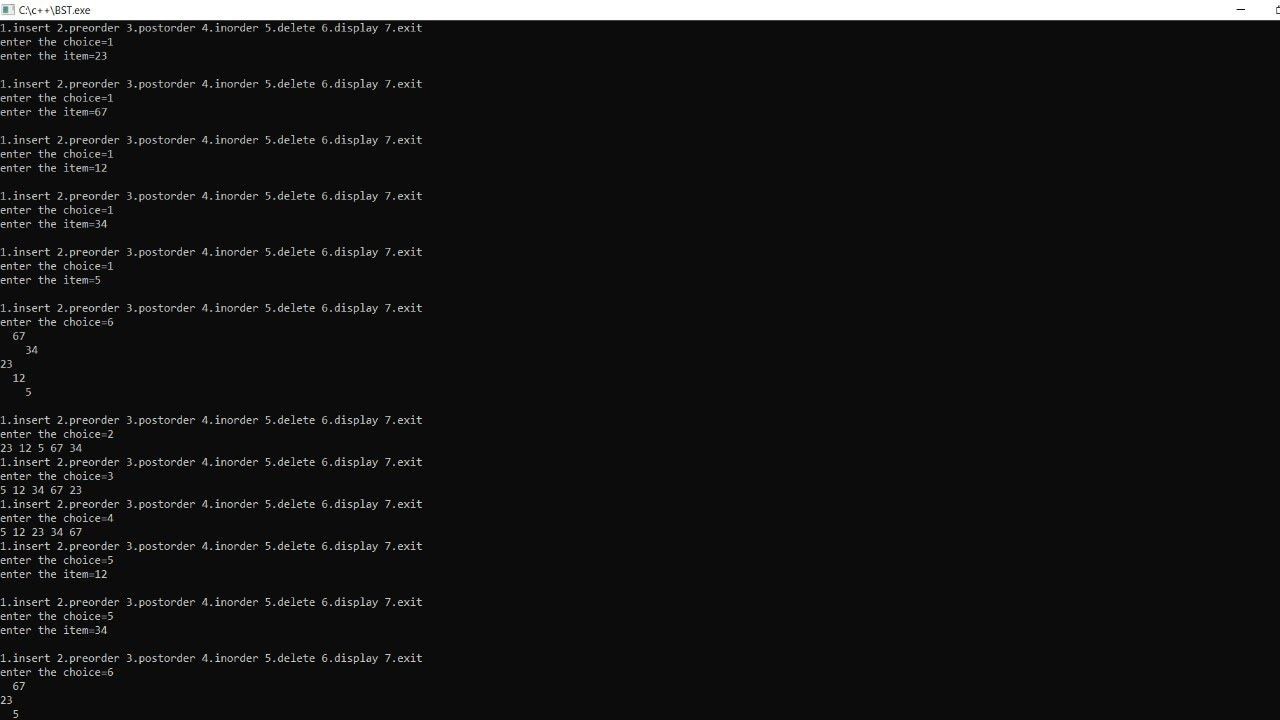
break; case 6:display(root,0); break; default:exit(0); break;

}

}

}

**Output:**



**b) To traverse the tree using all the methods i.e., in-order,preorder and post order**

#include<stdio.h>

#include<conio.h> #include<process.h> struct node{ int info; struct node\*llink; struct node\*rlink;

};

typedef struct node\*NODE;

NODE getnode(){ NODE x; x=(NODE)malloc(sizeof(struct node)); if(x==NULL){ printf("memory not available"); exit(0); } return x;

} void freenode(NODE x){

free(x);

}

NODE insert(int item,NODE root){ NODE temp,cur,prev; char direction[10];

int i;

temp=getnode(); temp->info=item; temp->llink=NULL; temp->rlink=NULL; if(root==NULL) return temp; printf("give direction to insert="); scanf("%s",direction); prev=NULL;

cur=root; for(i=0;i<strlen(direction)&&cur!=NULL;i++){ prev=cur; if(direction[i]=='l') cur=cur->llink; else cur=cur->rlink; } if(cur!=NULL||i!=strlen(direction)){ printf("insertion not possible.\n"); freenode(temp); return(root);

}

if(cur==NULL){

if(direction[i-1]=='l') prev->llink=temp; else prev->rlink=temp; } return(root); } void preorder(NODE root){ if(root!=NULL){ printf("%d ",root->info); preorder(root->llink); preorder(root->rlink);

}

} void inorder(NODE root){ if(root!=NULL){ inorder(root->llink); printf("%d ",root->info); inorder(root->rlink);

}

} void postorder(NODE root){ if (root!=NULL){ postorder(root->llink); postorder(root->rlink); printf("%d ",root->info);

} } void display(NODE root,int i){

int j;

if(root!=NULL){ display(root->rlink,i+1); for (j=1;j<=i;j++) printf(" "); printf("%d\n",root->info); display(root->llink,i+1);

} } void main(){

NODE root=NULL;

int choice,i,item; for(;;){ printf("\n1.insert 2.preorder 3.inorder 4.postorder 5.display\n"); printf("enter the choice="); scanf("%d",&choice); switch(choice){ case 1: printf("enter the item="); scanf("%d",&item); root=insert(item,root); break; case 2: if(root==NULL){ printf("tree is empty");

} else { printf("the preorder traversal is\n"); preorder(root);

} break; case 3:if(root==NULL) { printf("tree is empty");

} else

{ printf("the inorder traversal is \n"); inorder(root);

}

break;

case 4:if (root==NULL){ printf("tree is empty");

} else

{ printf("the postorder traversal is\n"); postorder(root);

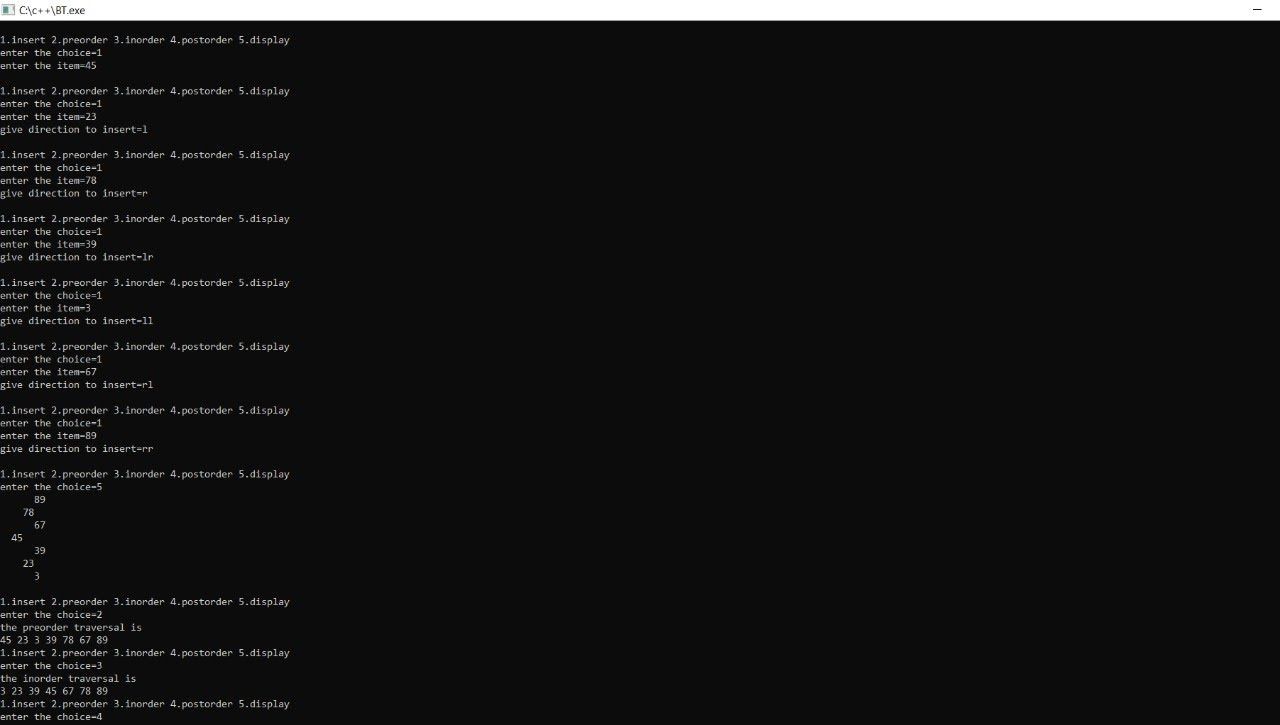
} break; case 5:display(root,1); break; default:exit(0);

}

}

}

**Output:**



***===============================================================***