**Data structure lab manual**

**CSE 208**



**Department offering the LAB course**

Computer Science and Engineering

**Course Pre-requisites**

CSE 103, CSE 201

**Course Description**

|  |
| --- |
| This is a laboratory course where students do practical exercise on the principles of programming languages with an emphasis on programming language implementation for various applications considering data structures. The purpose of this Laboratory course is to provide the students with solid foundations in the basic concepts of data structures with the practical works. |

**LAB Course Objectives**

1. To **emphasis** on the practical understanding of logical structures of data, their physical representation, design and analysis of algorithms operating on the structures, and techniques for program development and debugging;
2. To **demonstrate** the understanding and appropriate use of the data structures (e.g. arrays, linked lists, stack, queue, graph, and trees).

**Intended learning outcomes of the course (ILOs)**

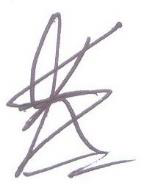
|  |  |
| --- | --- |
| SKILLS | **Solving practical problems using Data Structures in various applications** |
| 1. Critical analysis of situation and good programing knowledge and skills |
| 1. Design and develop solutions to real world problems |

**Mapping of Course LO and PLO:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Learning Outcome**  **(LO) of the Course** | **Program Learning Outcome (PLO)** | | | | | | | | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| **ILO 1** | MJ |  |  |  | MN |  |  |  | MN | MN |  |  |
| **ILO 2** | MJ |  |  |  | MN |  |  |  | MN | MN |  |  |

**Lab Instructions:**

* Strictly observe the instructions given by the Faculty / Lab. Instructor.
* NO FOOD, DRINK, IN ANY FORM is allowed in the lab.
* TURN OFF CELL PHONES! If you need to use it, please keep it in bags.
* Avoid all horseplay in the laboratory. Do not misbehave in the computer laboratory. Work quietly.
* Save often and keep your files organized.
* Don’t change settings and surf safely.
* Do not reboot, turn off, or move any workstation or PC.
* Do not load any software on any lab computer (without prior permission of Faculty and Technical Support Personnel). Only Lab Operators and Technical Support Personnel are authorized to carry out these tasks.
* Do not reconfigure the cabling/equipment without prior permission.
* Do not play games on systems.
* Violation of the above rules and etiquette guidelines will result in disciplinary action.



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature of the faculty

**ALGORITHM TO SEARCH AN ELEMENT USING LINEAR SEARCH**

1. Set k := 1 & loc : = 0
2. Repeat step 3 & 4 while loc : = 0 &k < = n
3. If (item = data[k]) loc : = k

Else

K = k + 1

1. If loc : = 0 ,then Print “no. not found” Else

Print “loc is the location of item”

1. Exit

**linear search**

#include<stdio.h>

#include<conio.h> void main()

{

int a[100],n,i,item,loc=-1; clrscr();

printf("\nEnter the number of element:"); scanf("%d",&n);

printf("Enter the number:\n"); for(i=0;i<=n-1;i++)

{

scanf("%d",&a[i]);

}

printf("Enter the no. to be search\n"); scanf("%d",&item);

for(i=0;i<=n-1;i++)

{

if(item==a[i])

{

loc=i; break;

}

}

if(loc>=0)

printf("\n%dis found in position%d",item,loc+1); else

printf("\nItem does not exits"); getch();

}

**ALGORITHM TO SEARCH AN ELEMENT USING BINARY SEARCH**

##### low = 1,high = n

1. Repeat step 3 to 5 while low <= high
2. mid = (low + high)
3. If a[mid] = x

Print “ found at mid” Return

1. If (a[mid] < x) low = mid + 1

Else

High = mid – 1

1. Print “x not found”
2. Exit

**//binary search**

#include<stdio.h>

#include<conio.h> void main()

{

int a[100],i,loc,mid,beg,end,n,flag=0,item; clrscr();

printf("How many elements"); scanf("%d",&n);

printf("Enter the element of the array\n"); for(i=0;i<=n-1;i++)

{

scanf("%d",&a[i]);

}

printf("Enter the element to be searching\n"); scanf("%d",&item);

loc=0; beg=0; end=n-1;

while((beg<=end)&&(item!=a[mid]))

{

mid=((beg+end)/2); if(item==a[mid])

{

printf("search is successfull\n"); loc=mid;

printf("position of the item%d\n",loc+1); flag=flag+1;

}

else

}

if(item<a[mid]) end=mid-1;

beg=mid+1;

if(flag==0)

{

printf("search is not successfull\n");

}

getch();

}

# ASSIGNMENT NO 3

#### Algorithm

Matmul(a,b,m,n,p) 1 for(i=1 to m)

2 for(j = 1 to p)

3 c[i][j] =0;

4 for(k= 1to n)

5 c[i][j] = c[i][j]+a[i][j]\*b[i][j] 6 exit

**matrix multiplication**

#include<stdio.h>

#include<conio.h> void main( )

{

int a[2][2], b[2][2],s[2][2];

int i,j,k;

clrscr();

printf("Enter first matrix:\n" ); for( i=1;i<=2;i++)

{

for( j=1;j<=2;j++)

{

printf("Enter%d%d\n",i,j); scanf("%d",&a[i][j]);

}

}

printf("Enter second matrix:\n"); for(i=1;i<=2;i++)

{

for(j=1;j<=2;j++)

{

printf("Enter %d%d\n",i,j); scanf("%d",&b[i][j]);

}

}

for(i=1;i<=2;i++)

{

for(j=1;j<=2;j++)

{ s[i][j]=0;

for(k=1;k<=2;k++)

{

s[i][j] =s[i][j]+a[i][k]\*b[k][j];

}

}

}

printf("Matrix Multiplication Is: \n"); for(i=1;i<=2;i++)

{

for (j=1;j<=2;j++)

{

printf("%d\n",s[i][j]);

}

}

getch();

}

#### Algorithm

Matadd(a,b,m,n) 1 for (i=1 to m

2 for(j= 1 to n)

3c[i][j] = a[i][j]+b[i][j] 4 exit

#### matrix addition

#include<stdio.h>

#include<conio.h> void main ( )

{

int a[2][2],b[2][2],s[2][2],i,j;

clrscr ();

printf("enter first matrix: \n"); for ( i=1; i<=2; i++)

{

for ( j=1; j<=2; j++)

{

printf("Enter %d%d", i,j, "element:");

scanf("%d",&a[i][j]);

}

}

printf("enter second matrix: \n"); for(i=1;i<=2;i++)

{

for(j=1; j<=2;j++)

{

printf( "enter %d%d",i + 1 ,j + 1 , "element:"); scanf("%d",&b[i][j]) ;

}

}

for (i=1;i<=2;i++)

{

for (j=1;j<=2;j++)

{

s[i][j]= a[i][j]+b[i][j];

}

}

printf("The addition matrix is:\n"); for (i=1;i<=2;i++)

{

for (j=1;j<=2;j++)

{

printf("%d\n",s[i][j] );

}

}

getch ();

}

###### Algorithm

Transpose(a,m,n)

1. for(i= 1 to m) for(j= 1 to n) b[i][j]= a[j][i]
2. for (i=1to m) for (j= 1to n) a[i][j]= b[i][j]

exit

#### transpose of a matrix

#include<stdio.h>

# include<conio.h> void main()

{

int a[10][10],b[10][10],i,j,m,n;

clrscr();

printf("Enter the order of the matrix\n"); printf("No. of rows : \n");

scanf("%d",&n);

printf("No. of columns :\n "); scanf("%d",&m);

printf("Enter the matrix elements\n"); for(i=0;i<=n-1;i++)

{

for(j=0;j<=m-1;j++)

{

scanf("%d\n",&a[i][j]);

b[j][i] = a[i][j];

}

}

printf("Matrix A was\n "); for(i=0;i<=n-1;i++)

{

for(j=0;j<=m-1;j++)

{

printf("%d\n",a[i][j]);

}

}

printf("Transpose of matrix A is \n"); for(i=0;i<=m-1;i++)

{

for(j=0;j<=n-1;j++)

{

printf("%d\n",b[i][j]);

}

}

getch( );

}

# ASSIGNMENT NO 4

**ALGORITHM TO SORT ARRAY USING QUICK SORT**

##### low =l, high = h, key a[(l+h)/2]

1. Repeat through step 7 while (low <= high)
2. Repeat step 4 while (a[low] < key)
3. low = low +1
4. Repeat step 6 while (a[high] > key)
5. high = high – 1
6. If (low <= high)
   1. temp = a[low]
   2. a[low] = a[high]
   3. a[high] = temp
   4. low = low + 1
   5. high = high + 1
7. If (l < high) quicksort (a,l,high)
8. If (h>low) quicksort (a,low,h)
9. Exit

**quick sort**

#include<stdio.h>

#include<conio.h>

#define max 100 int a[max],n,i,l,h; void main()

{

void input(void); input();

getch();

}

void input(void)

{

void output(int a[],int n);

void quick\_sort(int a[],int l,int h); printf("How many elements in the array : "); scanf("%d",&n);

printf("\n");

printf("Enter the elemennts : \n"); for(i=0;i<=n-1;i++)

{

scanf("%d",&a[i]);

} l=0;

h=n-1; quick\_sort(a,l,h);

printf("Sorted Array :\n "); output(a,n);

}

void quick\_sort(int a[],int l, int h)

{

int temp,key,low,high; low=l;

high=h; key=a[(low+high)/2]; do

{

while(key>a[low])

{

low++;

}

while(key<a[high])

{

high--;

}

if(low<=high)

{

temp=a[low]; a[low++]=a[high]; a[high--]=temp;

}

} while(low<=high); if(l<high) quick\_sort(a,l,high); if(low<h) quick\_sort(a,low,h);

}

void output(int a[],int n)

{

for(i=0;i<=n-1;i++)

{

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

}

}

**ASSIGNMENT NO 5 ALGORITHM TO IMPLEMENT BINARY SEARCH TREE**

INSERTION

1. t = newnode
2. t info = n
3. t left = t right = NULL
4. If (root = NULL) root = t

return

1. ptr = root
2. Repeat step 7 until ptr = NULL
3. If (ptr info > n)

If (ptr left = NULL) Ptr left = t

Return Else

Ptr = ptr left Else

If (ptr right = NULL) Ptr right = t

Return Else

Ptr = ptr right

DELETION

1. If (root = NULL) Print “Empty tree “ Return
2. ptr = root, par = NULL
3. Repeat step 4 & 5 until (ptr info = n or ptr = NULL)
4. par = ptr
5. If (ptrinfo > n) ptr = ptr left

Else

Ptr = ptr right

1. If ptr = NULL

print “ no. not present”

**//BST**

#include<stdio.h>

#include<conio.h> struct rec

{

long num;

struct rec \*left; struct rec \*right;

};

struct rec \*tree,\*second,\*head;

struct rec \*insert(struct rec \*tree,long num); struct rec \*copy(struct rec \*tree);

void inorder(struct rec \*tree); main()

{

int choice; long digit; do

{

choice=select(); switch(choice)

{

case 1:puts("Enter integers:To quit enter 0"); scanf("%ld",&digit); while(digit!=0)

{

tree=insert(tree,digit); scanf("%ld",&digit);

}continue;

case 2: copy(tree);continue;

case 3: puts("Inorder traversing TREE"); inorder(tree);continue;

case 4: puts("END");exit(0);

}

}while(choice!=4);

}

int select()

{

int selection; do

{

puts("Enter 1: Insert a node in the BST"); puts("Enter 2: Copy a tree to another BST"); puts("Enter 3: Display(inorder)the BST"); puts("Enter 4: END");

puts("Enter your choice"); scanf("%d",&selection); if((selection<1)||(selection>4))

{puts("Wrong choice: Try again"); getchar();

}

}while((selection<1)||(selection>4)); return selection;

}

struct rec \*insert(struct rec \*tree,long digit)

{

if(tree==NULL)

{

tree=(struct rec \*)malloc(sizeof(struct rec)); tree->left=tree->right=NULL;

tree->num=digit;

}

else

if(digit<tree->num)

tree->left=insert(tree->left,digit); else if(digit>tree->num)

tree->right=insert(tree->right,digit); else if(digit==tree->num)

{puts("Duplicate nodes: program exited");exit(0);

}

return(tree);

}

struct rec \*copy(struct rec \*tree)

{

second=(struct rec \*)malloc(sizeof(struct rec)); head=second;

if(tree!=NULL)

{

second->num=tree->num; if(tree->left!=NULL)

{

second->left->num=tree->left->num; copy(tree->right);

}

if(tree->right!=NULL)

{

second->right->num=tree->num; copy(tree->left);

}

}

return(head);

}

void inorder(struct rec \*tree)

{

if(tree!=NULL)

{

inorder(tree->left); printf("%12ld\n",tree->num); inorder(tree->right);

}

}

# ASSIGNMENT NO 6

**ALGORITHM TO IMPLEMENT LINKED LIST**

1. t = newmode( )
2. Enter info to be inserted
3. Read n
4. t info = n
5. t next = start
6. Start = t

INSERTION BEGIN

1. t next = start
2. start = t Return

MIDDLE

1. Enter info of the node after which new node to be inserted
2. Read x
3. p = start
4. Repeat step 5 until p info < > x
5. p = p next
6. t next = p next
7. p next = t
8. Return

LAST

1. p = start
2. Repeat step 3 until p next NULL
3. p = p next
4. t next = NULL
5. p next = t
6. Return

DELETION BEGIN

1. x = start
2. start = start next
3. delnode(x)

MIDDLE

1. Enter the info of node to be deleted
2. Read n
3. p = start
4. c = start
5. while (c info < > NULL) p = c

c = c next

1. p next = c next
2. delnode ( c )
3. Return LAST
4. p = start c = start
5. while (cnext < > NULL) p = c

c = cnext

1. p next = c next
2. delnode ( c)
3. Return

TRAVERSAL

1. p = start
2. while (p < > NULL) Print p info

P = p next

1. Return

##### // linked list//

#include<stdio.h>

#include<conio.h>

#include<malloc.h> struct node

{

int info;

struct node \*next;

};

typedef struct node NODE; NODE \*start;

void createmptylist(NODE \*\*start)

{

\*start=(NODE \*)NULL;

}

void traversinorder(NODE \*start)

{

while(start != (NODE \*) NULL)

{

printf("%d\n",start->info); start=start->next;

}

}

void insertatbegin(int item)

{

NODE \*ptr;

ptr=(NODE \*)malloc(sizeof(NODE)); ptr->info=item;

if(start==(NODE \*)NULL) ptr->next=(NODE \*)NULL; else

ptr->next=start; start=ptr;

}

void insert\_at\_end(int item)

{

NODE \*ptr,\*loc;

ptr=(NODE \*)malloc(sizeof(NODE)); ptr->info=item;

ptr->next=(NODE \*)NULL; if(start==(NODE\*)NULL) start=ptr;

else

{

loc=start;

while(loc->next!=(NODE \*)NULL) loc=loc->next;

loc->next=ptr;

}

}

void insert\_spe(NODE \*start,int item)

{

NODE \*ptr,\*loc; int temp,k;

for(k=0,loc=start;k<temp;k++)

{

loc=loc->next; if(loc==NULL)

{

printf("node in the list at less than one\n"); return;

}

}

ptr=(NODE \*)malloc(sizeof(NODE)); ptr->info=item;

ptr->next=loc->next;; loc->next=ptr;

}

void main()

{

int choice,item,after; char ch;

clrscr(); createmptylist(start); do

{

printf("1.Insert element at begin \n"); printf("2. insert element at end positon\n"); printf("3. insert specific the position\n"); printf("4.travers the list in order\n"); printf("5. exit\n");

printf("enter your choice\n"); scanf("%d",&choice); switch(choice)

{

case 1: printf("Enter the item\n"); scanf("%d",&item); insertatbegin(item); break;

case 2: printf("Enter the item\n"); scanf("%d",&item);

insert\_at\_end(item); break;

case 3: printf("Enter the item\n"); scanf("%d",&item); insert\_spe(start,item); break;

case 4: printf("\ntravers the list\n"); traversinorder(start); break;

case 5: return;

}

fflush(stdin);

printf("do your want continous\n"); scanf("%c",&ch);

}while((ch='y')||(ch='y')); getch();

}

**ASSIGNMENT NO 7 ALGORITHM TO IMPLEMENT DOUBLE LINKED LIST**

1. t = new node
2. Enter “the info to be inserted”
3. Read n
4. t info = n
5. t next = NULL
6. t prev NULL

INSERTION BEGIN

1. If start = NULL start = t
2. else

t next = NULL

t next prev = t start = t

Return

MIDDLE

1. Print “ enter info of the node after which you want to insert”
2. Read x
3. p = start
4. Repeat while p< > NULL If (pinfo = n)

tnext = pnext pnext = t

t prev = p

p nextprev = t Return

Else

P = pnext

1. Print x not found

tnext = NULL pnext = t

DELETION BEGIN

1. p = start
2. pnextprev = NULL
3. start = pnext
4. start = pnext
5. delnode(p)
6. Return

MIDDLE

1. Enter “info of the node to be deleted”
2. Read x
3. p = start
4. Repeat until p< > NULL

If(pinfo = x) pprevnext = pnext pnext prev = pprev delnode(p)

Return Else

P = pnext

1. Print “x not found”

LAST

1. P = start
2. Repeat while p< > NULL If(pnext = NULL) Delnode(p)
3. Return

DISPLAY

1. p = start
2. Repeat while p < > NULL Print pinfo

P = p next

#include<stdio.h>

#include<conio.h> int select(); struct rec

{

char name[80]; struct rec \*next;

};

struct rec \*rear;

struct rec \*create(struct rec \*list); struct rec \*insert1(struct rec \*node); struct rec \*insert2(struct rec \*node); struct rec \*insert3(struct rec \*node); struct rec \*insert4(struct rec \*node); struct rec \*delete(struct rec \*node); void \*display(struct rec \*list);

int nodes; main()

{

struct rec \*first=NULL; int choice;

clrscr(); do

{

choice=select(); switch(choice)

{

case 1: first=create(first);continue;

case 2: first=insert1(first);continue;

case 3: first=insert2(first);continue;

case 4: first=insert3(first);continue;

case 5: first=insert4(first);continue;

case 6: first=delete(first);continue;

case 7: display(first);continue;

case 8: puts("END");exit(0);

}

}while(choice!=8);

}

int select()

{

int selection; do

{

puts("Enter 1: create the list");

puts("Enter 2: insert in the beginnig of the list"); puts("Enter 3: insert after a node in the list"); puts("Enter 4: insert before a node in the list"); puts("Enter 5: insert in the end of the list"); puts("Enter 6: delete the list");

puts("Enter 7: display the list"); puts("Enter 8: END");

puts("Enter your choice"); scanf("%d",&selection);

}while(selection<1||selection>8); return selection;

}

struct rec \*create(struct rec \*first)

{

struct rec \*element;

first=(struct rec\*)malloc(sizeof(struct rec)); puts("Enter/name/word/text: To quit enter\*"); scanf(" %[^\n]",first->name);

first->next=first; rear=first;

rear->next=first;for(;;)

{

element=(struct rec\*)malloc(sizeof(struct rec)); scanf(" %[^\n]",element->name); if(strcmp(element->name,"\*")==0)break;

element->next=first; rear->next=element; rear= element;

}

return(first);

}

struct rec \*insert1(struct rec \*first)

{

struct rec \*node;

node=(struct rec\*)malloc(sizeof(struct rec)); puts("Enter node/name to be inserted"); scanf(" %[^\n]",node->name);

if(first==NULL)

{

node->next=first; rear=first;

}

else

{

node->next=first; first=node;

rear->next=first;

}

return(first);

}

struct rec \*insert2(struct rec \*first)

{

struct rec \*current,\*prior,\*x; struct rec \*node;current=first;

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

puts("Enter node/name after which new node to be inserted"); scanf(" %[^\n]\n",node->name);

x=(struct rec\*)malloc(sizeof(struct rec)); puts("Enter node/name to be inserted"); scanf(" %[^\n]",x->name); while(current!=rear && current!=NULL)

{

if(strcmp(current->name,node->name)==0)

{

x->next=current->next; current->next=x; return(first);

}

else current=current->next;

}

if(strcmp(current->name,node->name)==0)

{

x->next=first; rear->next=x; rear=x; return(first);

}

puts("Node does not exist in the list"); return(first);

}

struct rec \*insert3(struct rec \*first)

{

struct rec \*node,\*current,\*x,\*prior; current=first;

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

puts("Enter node/name before which new node to be inserted"); scanf(" %[^\n]",node->name);

x=(struct rec\*)malloc(sizeof(struct rec)); puts("Enter node/name to be inserted"); scanf(" %[^\n]",x->name); if(strcmp(current->name,node->name)==0)

{

x->next=first; first=x; return(first);

}

while(current!=NULL)

{

prior=current; current=current->next;

if(strcmp(current->name,node->name)==0)

{

x->next=current; prior->next=x;

return(first);

}

}

puts("Node does not exist in the list"); return(first);

}

struct rec \*insert4(struct rec \*first)

{

struct rec \*element;

element=(struct rec\*)malloc(sizeof(struct rec)); puts("Enter node/name to be inserted at the end of list"); scanf(" %[^\n]",element->name);

element->next=first; rear->next=element; rear=element; return(first);

}

struct rec \*delete(struct rec \*first)

{

struct rec \*current,\*prior,\*node; current=first;

node=(struct rec\*)malloc(sizeof(struct rec)); puts("Enter node/name to be delete");

scanf(" %[^\n]",node->name); if(strcmp(current->name,node->name)==0)

{

first=current->next; rear->next=first; free(current); return(first);

}

while(current!=rear && current!=NULL)

{

prior=current; current=current->next;

if(strcmp(current->name,node->name)==0)

{

prior->next=current->next; free(current); return(first);

}

}

if(strcmp(current->name,node->name)==0)

{

prior->next=current->next; prior->next=first; rear=prior;

free(current); return(first);

}

puts("Node does not exist in the list"); return(first);

}

void \*display(struct rec \*first)

{

int node=0;

{

node++;

printf("%s\n",first->name); first=first->next;

}

while((first!=rear->next)&&(first!=NULL)); printf("Nuber of nodes= %d\n",node);

}

**ASSIGNMENT NO 8 ALGORITHM TO IMPLEMENT QUEUE AS LINKED LIST**

CREATE

1. t = new node
2. Enter info to be inserted
3. Read n
4. t info = n
5. t next = front
6. front = t

INSERTION

1. r next = t
2. t next = NULL
3. Return

DELETION

1. x = front
2. front = front next
3. delnode(x)
4. Return

DISPLAY

1. If (front = NULL) Print “ empty queue” Return

Else

P = start

Repeat until (p< > NULL) Print p info

P = pnext Return

**Program**

#include<stdio.h>

#include<conio.h> struct queue

{

int no;

struct queue \*next;

}

\*start=NULL; void add(); int del();

void traverse(); void main()

{

int ch; char choice;

{

clrscr();

printf("----1. add\n");

printf("----2. delete\n");

printf("----3. traverse\n");

printf("----4. exit\n"); printf("Enter your choice\n"); scanf("%d",&ch);

switch(ch)

{

case 1: add(); break;

case 2: printf("the delete element is\n%d",del()); break;

case 3: traverse(); break;

case 4: return;

default : printf("wrong choice\n");

};

fflush(stdin); scanf("%c",&choice);

}

while(choice!=4);

}

void add()

{

struct queue \*p,\*temp; temp=start;

p=(struct queue\*)malloc(sizeof(struct queue)); printf("Enter the data");

scanf("%d",&p->no); p->next=NULL; if(start==NULL)

{

start=p;

}

else

{

while(temp->next!=NULL)

{

temp=temp->next;

}

temp->next=p;

}

}

int del()

{

struct queue \*temp; int value; if(start==NULL)

{

printf("queue is empty"); getch();

return(0);

}

else

{

temp=start; value=temp->no; start=start->next; free(temp);

}

return(value);

}

void traverse()

{

struct queue \*temp; temp=start;

while(temp->next!=NULL)

{

printf("no=%d",temp->no); temp=temp->next;

}

printf("no=%d",temp->no); getch();

}

# ASSIGNMENT NO 9

**ALGORITHM TO IMPLEMENT STACK USING ARRAY**

INSERTION

PUSH(item)

1. If (item = max of stack) Print “overflow” Return
2. top = top + 1
3. stack[top] = item
4. Return

DELETION

POP(item)

1. If (top = - 1) Print “underflow” Return
2. Item = stack[top]
3. top = top – 1
4. Return

DISPLAY

1. If top = - 1

Print “underflow”

1. repeat step 3 for i = top to i >= 0
2. Print stack[i]
3. Return

#include<stdio.h>

#include<conio.h>

#define MAXSIZE 10 void push();

int pop();

void traverse(); int stack[MAXSIZE]; int Top=-1;

void main()

{

int choice; char ch; do

{

clrscr(); printf("\n1. PUSH ");

printf("\n2. POP "); printf("\n3. TRAVERSE "); printf("\nEnter your choice"); scanf("%d",&choice); switch(choice)

{

case 1: push(); break;

case 2: printf("\nThe deleted element is %d",pop()); break;

case 3: traverse(); break;

default: printf("\nYou Entered Wrong Choice");

}

printf("\nDo You Wish To Continue (Y/N)"); fflush(stdin);

scanf("%c",&ch);

}

while(ch=='Y' || ch=='y');

}

void push()

{

int item;

if(Top == MAXSIZE - 1)

{

printf("\nThe Stack Is Full"); getch();

exit(0);

}

else

{

printf("Enter the element to be inserted"); scanf("%d",&item);

Top= Top+1; stack[Top] = item;

}

}

int pop()

{

int item; if(Top == -1)

{

printf("The stack is Empty"); getch();

exit(0);

}

else

{

item = stack[Top]; Top = Top-1;

}

return(item);

}

void traverse()

{

int i;

if(Top == -1)

{

printf("The Stack is Empty"); getch();

exit(0);

}

else

{

for(i=Top;i>=0;i--)

{

printf("Traverse the element"); printf("\n%d",stack[i]);

}

}

}

# ASSIGNMENT NO 10

**ALGORITHM TO IMPLEMENT STACK AS LINKED LIST**

PUSH( )

1. t = newnode( )
2. Enter info to be inserted
3. Read n
4. tinfo = n
5. tnext = top
6. top = t
7. Return

POP( )

1. If (top = NULL) Print “ underflow” Return
2. x = top
3. top = top next
4. delnode(x)
5. Return

##### // stack using linked list//

#include<stdio.h>

#include<conio.h> struct stack

{

int no;

struct stack \*next;

}

\*start=NULL;

typedef struct stack st; void push();

int pop(); void display(); void main()

{

char ch;

int choice,item; do

{

clrscr();

printf("\n 1: push");

printf("\n 2: pop");

printf("\n 3: display"); printf("\n Enter your choice"); scanf("%d",&choice);

switch (choice)

{

case 1: push(); break;

case 2: item=pop();

printf("The delete element in %d",item); break;

case 3: display(); break;

default : printf("\n Wrong choice");

};

printf("\n do you want to continue(Y/N)"); fflush(stdin);

scanf("%c",&ch);

}

while (ch=='Y'||ch=='y');

}

void push()

{

st \*node;

node=(st \*)malloc(sizeof(st));

printf("\n Enter the number to be insert"); scanf("%d",&node->no);

node->next=start; start=node;

}

int pop()

{

st \*temp; temp=start; if(start==NULL)

{

printf("stack is already empty"); getch();

exit();

}

else

{

start=start->next; free(temp);

}

return(temp->no);

}

void display()

{

st \*temp; temp=start;

while(temp->next!=NULL)

{

printf("\nno=%d",temp->no); temp=temp->next;

}

printf("\nno=%d",temp->no);

}

# ASSIGNMENT NO 11

## ALGORITHM TO SORT ARRAY USING BUBBLE SORT

##### Repeat steps 2 & 3 for k = 1 to N-1

1. Set ptr =1
2. Repeat while ptr <= N-k
3. (a) If data[ptr] > data[ptr + 1],then Interchange data[ptr] and data[ptr + 1]

(b) ptr = ptr + 1

1. Exit

**bubble sort**

#include<stdio.h>

#include<conio.h> void main()

{

int a[100],n,i,j,temp; clrscr();

printf("How many elements"); scanf("%d",&n);

printf("Enter the element of array"); for(i=0;i<=n-1;i++)

{

scanf("%d",&a[i]);

}

for(i=0;i<=n-1;i++)

{

for(j=0;j<=n-1-i;j++)

{

if(a[j]>a[j+1])

{

temp=a[j]; a[j]=a[j+1]; a[j+1]=temp;

}

}

}

printf("Element of array after the sorting are:\n"); for(i=0;i<=n-1;i++)

{

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

}

getch();

}

**ASSIGNMENT NO 12**

**Algorithm for tree traversal**

Preorder(root)

If root = null then exit Process root->info Preorder root->left; Preorder root->right Exit

Inorder(root)

If root = null then exit Inorder root->left Process root->info Inorder root->right

Exit

Postorder(root)

If root = null then exit Postorder root->left Postorder root->right Postorder root->info exit

// traversing a tree

#include<stdio.h> struct rec

{

long num;

struct rec \*left; struct rec \*right;

};

struct rec \*tree=NULL;

struct rec \*insert(struct rec \*tree,long num); void preorder(struct rec \*tree);

void inorder(struct rec \*tree); void postorder(struct rec \*tree); int count=1;

main()

{

int choice; long digit; do

{

choice=select(); switch(choice)

{

case 1: puts("Enter integer: To quit enter 0"); scanf("%ld",&digit);

while(digit!=0)

{

tree=insert(tree,digit); scanf("%ld",&digit);

}continue;

case 2: puts("\npreorder traversing TREE"); preorder(tree);continue;

case 3: puts("\ninorder traversing TREEE"); inorder(tree);continue;

case 4: puts("\npostorder traversing TREE"); postorder(tree);continue;

case 5: puts("END");exit(0);

}

}while(choice!=5);

}

int select()

{

int selection; do

{

puts("Enter 1: Insert a node in the BT"); puts("Enter 2: Display(preorder)the BT"); puts("Enter 3: Display(inorder)the BT"); puts("Enter 4: Display(postorder)the BT"); puts("Enter 5: END");

puts("Enter your choice"); scanf("%d",&selection);

if((selection<1)||(selection>5))

{

puts("wrong choice:Try again"); getch(); }

}while((selection<1)||(selection>5)); return (selection);

}

struct rec \*insert(struct rec \*tree,long digit)

{

if(tree==NULL)

{

else

tree=(struct rec \*)malloc(sizeof(struct rec)); tree->left=tree->right=NULL;

tree->num=digit;count++;

}

else

}

if(count%2==0)

tree->left=insert(tree->left,digit);

tree->right=insert(tree->right,digit); return(tree);

void preorder(struct rec \*tree)

{

if(tree!=NULL)

{

printf("%12ld\n",tree->num); preorder(tree->left); preorder(tree->right);

}

}

void inorder(struct rec \*tree)

{

if(tree!=NULL)

{

inorder(tree->left); printf("%12ld\n",tree->num); inorder(tree->right);

}

}

void postorder(struct rec \*tree)

{

if(tree!=NULL)

{

postorder(tree->left); postorder(tree->right); printf("%12ld\n",tree->num);

}

}

int select()

{

int selection; do

{

puts("Enter 1: Insert a node in the BT"); puts("Enter 2: Display(preorder)the BT"); puts("Enter 3: Display(inorder)the BT"); puts("Enter 4: Display(postorder)the BT"); puts("Enter 5: END");

puts("Enter your choice"); scanf("%d",&selection);

if((selection<1)||(selection>5))

{

puts("wrong choice:Try again"); getch(); }

}while((selection<1)||(selection>5)); return (selection);

}

struct rec \*insert(struct rec \*tree,long digit)

{

if(tree==NULL)

{

else

tree=(struct rec \*)malloc(sizeof(struct rec)); tree->left=tree->right=NULL;

tree->num=digit;count++;

}

else

}

if(count%2==0)

tree->left=insert(tree->left,digit);

tree->right=insert(tree->right,digit); return(tree);

void preorder(struct rec \*tree)

{

if(tree!=NULL)

{

printf("%12ld\n",tree->num); preorder(tree->left); preorder(tree->right);

}

}

void inorder(struct rec \*tree)

{

if(tree!=NULL)

{

inorder(tree->left); printf("%12ld\n",tree->num); inorder(tree->right);

}

}

void postorder(struct rec \*tree)

{

if(tree!=NULL)

{

postorder(tree->left); postorder(tree->right); printf("%12ld\n",tree->num);

}

}

**FAQ for data structure lab**

##### 1) what is the advantage of binary search method over linear search. 2)what are the drawbacks of binary search method.

3)Calculate the complexity of sorting methods. 4)compare various sorting methods by their performance.

1. What is the advantage of dynamic implementation over static implementation of stack.
2. application of stack.
3. Advantage of circular queue over linear queue. 8)Drawback of static implementation of queue 9)Explain the use of stack in tree traversal.

10)What is the advantage of BST and where it is used. 11)What is the advantage of doubly linked list.