Data Structures and Algorithms

1. Program to display array elements in reverse order

#include<iostream.h>

void main()

{

int i, a[5]={10, 20, 30, 40, 50};

clrscr();

for(i=5;i>0;i--)

cout<<”a[“<<i<<”]=”<<a[i]<<endl;

}

2. Program on how to create array, store elements into array and display elements from array

#include<iostream.h>

#include<conio.h>

# define max 5

main()

{

int arr[max],i;

clrscr();

cout<<"Enter elements into array (maximum 5)";

for(i=0;i<max;i++)

cin>>arr[i];

cout<<"Your one dimensional array is"<<endl;

for(i=0;i<max;i++)

cout<<"arr["<<i<<"]="<<arr[i]<<endl);

getch();

}

3. Program to store and calculate the sum of 5 numbers entered by the user using arrays #include <iostream> using namespace std; int main() { int numbers[5], sum = 0; cout << "Enter 5 numbers: "; for (int i = 0; i < 5; ++i) { cin >> numbers[i]; sum += numbers[i]; } cout << "Sum = " << sum << endl; return 0; }

4. Program on how to create Two Dimensional Array

#include<iostream.h>

#include<conio.h>

#define row 3

#define col 3

main()

{

int arr[row][col],i,j;

clrscr();

cout<<"Enter values in two dimensional arrays(3 rows, 3 columns) "<<endl;

for(i=0;i<row;i++)

for(j=0;j<col;j++)

cin>>arr[i][j]);

cout<<"your two dimensional array is<<endl;

for(i=0;i<row;i++)

for(j=0;j<col;j++)

cout<<arr[i][j]<<"\t";

cout<<endl;

}

return 0;

getch();

}

5. Program to display all elements of an initialised two dimensional array #include <iostream> using namespace std; int main() { int test[3][2] = { {2, -5}, {4, 0}, {9, 1} }; for(int i = 0; i < 3; ++i) for(int j = 0; j < 2; ++j) cout<< "test[" << i << "][" << j << "] = " << test[i][j] << endl; return 0; }

6. Program to Calculate Average of Numbers Using Arrays #include <iostream> using namespace std; int main() { int n, i; float num[100], sum=0.0, average; cout << "Enter the numbers of data: ";

cin >> n; while (n > 100 || n <= 0) { cout << "Error! number should in range of (1 to 100)." << endl; cout << "Enter the number again: "; cin >> n; } for(i = 0; i < n; ++i) { cout << i + 1 << ". Enter number: "; cin >> num[i]; sum += num[i]; } average = sum / n; cout << "Average = " << average; return 0; }

7. Program to Display Largest Element of an array #include <iostream> using namespace std; int main() {

int i, n; float arr[100]; cout << "Enter total number of elements(1 to 100): "; cin >> n; cout << endl; for(i = 0; i < n; ++i) { cout << "Enter Number " << i + 1 << " : "; cin >> arr[i]; } for(i = 1;i < n; ++i) { if(arr[0] < arr[i]) arr[0] = arr[i]; } cout << "Largest element = " << arr[0]; return 0; }

8. Program to Find Transpose of a Matrix #include <iostream> using namespace std; int main() { int a[10][10], trans[10][10], r, c, i, j; cout << "Enter rows and columns of matrix: "; cin >> r >> c; cout << endl << "Enter elements of matrix: " << endl;

for(i = 0; i < r; ++i) for(j = 0; j < c; ++j) { cout << "Enter elements a" << i + 1 << j + 1 << ": "; cin >> a[i][j]; } cout << endl << "Entered Matrix: " << endl; for(i = 0; i < r; ++i) for(j = 0; j < c; ++j) { cout << " " << a[i][j]; if(j == c - 1) cout << endl << endl; } for(i = 0; i < r; ++i) for(j = 0; j < c; ++j) { trans[j][i]=a[i][j]; } cout << endl << "Transpose of Matrix: " << endl; for(i = 0; i < c; ++i) for(j = 0; j < r; ++j) { cout << " " << trans[i][j]; if(j == r - 1) cout << endl << endl; } return 0; }

9. Program to store temperature of two different cities for a week and display it #include <iostream> using namespace std; const int CITY = 2; const int WEEK = 7; int main() { int temperature[CITY][WEEK]; cout << "Enter all temperature for a week of first city and then second city. \n"; for (int i = 0; i < CITY; ++i) { for(int j = 0; j < WEEK; ++j) { cout <<"City"<< i + 1 << ", Day " << j + 1 << " : "; cin >> temperature[i][j]; } } cout << "\n\nDisplaying Values:\n"; for (int i = 0; i < CITY; ++i)

for(int j = 0; j < WEEK; ++j) cout << "City " << i + 1 << ", Day " << j + 1 << " = " << temperature[i][j] << endl; return 0; }

10. Program to Store value entered by user in three dimensional array and display it #include <iostream> using namespace std; int main() { int test[2][3][2]; cout << "Enter 12 values: \n"; for(int i = 0; i < 2; ++i) for (int j = 0; j < 3; ++j) for(int k = 0; k < 2; ++k ) cin >> test[i][j][k];

cout<<"\nDisplaying Value stored:"<<endl; for(int i = 0; i < 2; ++i) for (int j = 0; j < 3; ++j) for(int k = 0; k < 2; ++k) cout << "test[" << i << "][" << j << "][" << k << "] = " << test[i][j][k] << endl; return 0; }

11. Program to reverse the element of an integer 1-D array

#include<iostream.h> #include<conio.h> void main() { int arr[50], size, i, j, temp; cout<<"Enter array size : "; cin>>size; cout<<"Enter array elements : "; for(i=0; i<size; i++) cin>>arr[i]; j=i-1;

i=0;

while(i<j) { temp=arr[i]; arr[i]=arr[j]; arr[j]=temp; i++; j--; } cout<<"Now the Reverse of the Array is : \n"; for(i=0; i<size; i++)

cout<<arr[i]<<" "; getch(); }

12. Program to Subtract Two Matrices #include<iostream.h> #include<conio.h> void main() { int arr1[3][3], arr2[3][3], arr3[3][3], sub, i, j; cout<<"Enter 3\*3 Array 1 Elements : "; for(i=0; i<3; i++) for(j=0; j<3; j++) cin>>arr1[i][j]; cout<<"Enter 3\*3 Array 2 Elements : "; for(i=0; i<3; i++) for(j=0; j<3; j++) cin>>arr2[i][j]; cout<<"Subtracting array (array1-array2) ... \n"; for(i=0; i<3; i++) for(j=0; j<3; j++) arr3[i][j]=arr1[i][j]-arr2[i][j]; cout<<"Result of Array1 - Array2 is :\n"; for(i=0; i<3; i++) for(j=0; j<3; j++) cout<<arr3[i][j]<<" "; cout<<"\n"; getch(); }

13. Program to Insert Element in Array #include<iostream.h> #include<conio.h> void main() { int arr[50], size, insert, i, pos; cout<<"Enter Array Size : "; cin>>size; cout<<"Enter array elements : "; for(i=0; i<size; i++) cin>>arr[i]; cout<<"Enter element to be insert : "; cin>>insert; cout<<"At which position (Enter index number) ? "; cin>>pos; for(i=size; i>pos; i--) arr[i]=arr[i-1]; arr[pos]=insert; cout<<"Element inserted successfully..!!\n"; cout<<"Now the new array is : \n"; for(i=0; i<size+1; i++)

cout<<arr[i]<<" "; getch(); }

14. Program to Delete Element from Array #include<iostream.h> #include<conio.h> void main() { int arr[50], size, i, del, count=0; cout<<"Enter array size : "; cin>>size; cout<<"Enter array elements : "; for(i=0; i<size; i++) cin>>arr[i]; cout<<"Enter element to be delete : "; cin>>del; for(i=0; i<size; i++) { if(arr[i]==del) { for(int j=i; j<(size-1); j++) arr[j]=arr[j+1]; count++; break; } } if(count==0) cout<<"Element not found..!!"; else { cout<<"Element deleted successfully..!!\n"; cout<<"Now the new array is :\n"; for(i=0; i<(size-1); i++) cout<<arr[i]<<" "; } getch(); }

15. Program to Merge Two Arrays #include<iostream.h> #include<conio.h> void main() { int arr1[50], arr2[50], size1, size2, size, i, j, k, merge[100]; cout<<"Enter Array 1 Size : "; cin>>size1; cout<<"Enter Array 1 Elements : "; for(i=0; i<size1; i++) cin>>arr1[i]; cout<<"Enter Array 2 Size : ";

cin>>size2; cout<<"Enter Array 2 Elements : "; for(i=0; i<size2; i++) cin>>arr2[i]; for(i=0; i<size1; i++) merge[i]=arr1[i]; size=size1+size2; for(i=0, k=size1; k<size && i<size2; i++, k++) merge[k]=arr2[i]; cout<<"Now the new array after merging is :\n"; for(i=0; i<size; i++) cout<<merge[i]<<" "; getch(); }

16. Program to swap first and last element of an integer 1-d array #include<iostream> using namespace std; int main() { int Arr[100],n,i,temp; cout<<"Enter number of elements you want to insert "; cin>>n; for(i=0;i<n;i++) { cout<<"Enter element "<<i+1<<":"; cin>>Arr[i]; } temp=Arr[0]; Arr[0]=Arr[n-1]; Arr[n-1]=temp; cout<<"\nArray after swapping"<<endl; for(i=0;i<n;i++) cout<<Arr[i]<<" "; return 0; }

17. Program to find the largest and smallest element of an array #include<iostream> using namespace std; int main() { int Arr[100],n,i,small,large; cout<<"Enter number of elements you want to insert "; cin>>n; for(i=0;i<n;i++) { cout<<"Enter element "<<i+1<<":";

cin>>Arr[i]; } small=Arr[0]; large=Arr[0]; for(i=1;i<n;i++) { if(Arr[i]<small) small=Arr[i]; if(Arr[i]>large) large=Arr[i]; } cout<<"\nLargest element is :"<<large; cout<<"\nSmallest element is :"<<small; return 0; }

18. Program to Add Two Matrices using Multidimensional Arrays #include <iostream> using namespace std; int main() { int r, c, a[100][100], b[100][100], sum[100][100], i, j; cout << "Enter number of rows (between 1 and 100): "; cin >> r; cout << "Enter number of columns (between 1 and 100): "; cin >> c; cout << endl << "Enter elements of 1st matrix: " << endl; for(i = 0; i < r; ++i) for(j = 0; j < c; ++j) { cout << "Enter element a" << i + 1 << j + 1 << " : "; cin >> a[i][j]; } cout << endl << "Enter elements of 2nd matrix: " << endl; for(i = 0; i < r; ++i) for(j = 0; j < c; ++j) { cout << "Enter element b" << i + 1 << j + 1 << " : "; cin >> b[i][j]; } for(i = 0; i < r; ++i) for(j = 0; j < c; ++j) sum[i][j] = a[i][j] + b[i][j]; cout << endl << "Sum of two matrix is: " << endl; for(i = 0; i < r; ++i) for(j = 0; j < c; ++j) { cout << sum[i][j] << " "; if(j == c - 1) cout << endl;

} return 0; }

19. Program to Multiply two matrices without using functions #include <iostream> using namespace std; int main() { int a[10][10], b[10][10], mult[10][10], r1, c1, r2, c2, i, j, k; cout << "Enter rows and columns for first matrix: "; cin >> r1 >> c1; cout << "Enter rows and columns for second matrix: "; cin >> r2 >> c2; while (c1!=r2) { cout << "Error! column of first matrix not equal to row of second.";

cout << "Enter rows and columns for first matrix: "; cin >> r1 >> c1;

cout << "Enter rows and columns for second matrix: "; cin >> r2 >> c2; } cout << endl << "Enter elements of matrix 1:" << endl; for(i = 0; i < r1; ++i)

for(j = 0; j < c1; ++j) { cout << "Enter element a" << i + 1 << j + 1 << " : "; cin >> a[i][j]; } cout << endl << "Enter elements of matrix 2:" << endl; for(i = 0; i < r2; ++i) for(j = 0; j < c2; ++j) { cout << "Enter element b" << i + 1 << j + 1 << " : "; cin >> b[i][j]; } for(i = 0; i < r1; ++i) for(j = 0; j < c2; ++j) mult[i][j]=0; for(i = 0; i < r1; ++i) for(j = 0; j < c2; ++j) for(k = 0; k < c1; ++k)

mult[i][j] += a[i][k] \* b[k][j]; cout << endl << "Output Matrix: " << endl; for(i = 0; i < r1; ++i) for(j = 0; j < c2; ++j) { cout << " " << mult[i][j]; if(j == c2-1) cout << endl; }

return 0; }

20. Program for addition of two polynomials #include<iostream.h> #include<iomanip.h> #include<conio.h> struct poly { int coeff; int pow; poly \*next; }; class add2poly { poly \*poly1, \*poly2, \*poly3; public: add2poly(){poly1=poly2=poly3=NULL;} void addpoly(); void display(); }; void add2poly :: addpoly() { int i,p; poly \*newl=NULL,\*end=NULL; cout<<"Enter highest power for x\n";

cin>>p; //Read first poly cout<<"\nFirst Polynomial\n"; for(i=p;i>=0;i--)

{ newl=new poly; newl->pow=p; cout<<"Enter Co-efficient for degree"<<i<<":: cin>>newl->coeff; newl->next=NULL; if(poly1==NULL) poly1=newl; else end->next=newl; end=newl; } //Read Second poly cout<<"\n\nSecond Polynomial\n"; end=NULL; for(i=p;i>=0;i--) { newl=new poly; newl->pow=p; cout<<"Enter Co-efficient for degree"<<i<<"::

cin>>newl->coeff; newl->next=NULL; if(poly2==NULL) poly2=newl; else end->next=newl; end=newl;

} //Addition Logic poly \*p1=poly1,\*p2=poly2; end=NULL; while(p1 !=NULL && p2!=NULL) { if(p1->pow == p2->pow) { newl=new poly; newl->pow=p--; newl->coeff=p1->coeff + p2->coeff; newl->next=NULL; if(poly3==NULL) poly3=newl; else end->next=newl; end=newl; } p1=p1->next; p2=p2->next; } } void add2poly :: display() {

poly \*t=poly3; cout<<"\n\nAnswer after addition is : "; while(t!=NULL) {

cout.setf(ios::showpos); cout<<t->coeff; cout.unsetf(ios::showpos); cout<<"X"<<t->pow; t=t->next; }

} void main() {

clrscr(); add2poly obj; obj.addpoly(); obj.display(); getch(); }

21. Code to PUSH Element in a stack using Array

void push() { int item; if(Top == MAXSIZE - 1) { printf("nThe Stack Is Full"); exit(0); } else { printf("Enter the element to be inserted "); scanf("%d",&item); Top= Top+1; stack[Top] = item; } }

22. Code to POP elements from stack using Array

int pop() { int item; if(Top == -1) { printf("The stack is Empty"); exit(0); } else { item = stack[Top]; Top = Top-1; } return(item); }

23. Code to traverse the stack using Array

void traverse() { int i; if(Top == -1) { printf("The Stack is Empty"); exit(0); } else { for(i=Top;i>=0;i--) { printf("Traverse the element "); printf("%dn",stack[i]);

} }

}

24. Code for the creation of new a node in a linked list

void createnode(int value) { node \*temp=new node; temp->data=value; temp->next=NULL; if(head==NULL) { head=temp; tail=temp; temp=NULL; } else { tail->next=temp; tail=temp; } }

25. Code for displaying nodes of linked list void display() { node \*temp=new node; temp=head; while(temp!=NULL) { cout<<temp->data<<"\t"; temp=temp->next; } }

26. Code to insert the node at the start of the linked list void insert\_start(int value) { node \*temp=new node; temp->data=value; temp->next=head; head=temp; }

27. Code to insert the node at the particular position in the linked list

void insert\_position(int pos, int value) { node \*pre=new node; node \*cur=new node;

node \*temp=new node; cur=head; for(int i=1;i<pos;i++) { pre=cur; cur=cur->next; } temp->data=value; pre->next=temp; temp->next=cur; }

28. Code to delete the first node from the linked list

void delete\_first() { node \*temp=new node; temp=head; head=head->next; delete temp; }

29. Code to delete the last node from the linked list

void delete\_last() { node \*current=new node; node \*previous=new node; current=head; while(current->next!=NULL) { previous=current; current=current->next; } tail=previous; previous->next=NULL; delete current; }

30. Code to delete the node from a particular position from a linked list

void delete\_position(int pos) { node \*current=new node; node \*previous=new node; current=head; for(int i=1;i<pos;i++) { previous=current; current=current->next; } previous->next=current->next; }

31. Program to calculate factorial of a given number using recursion in ‘C’ #include<iostream.h> #include<conio.h> void main() { int n,fact; int rec(int); clrscr(); cout<<"Enter the number:->"; cin>>n; fact=rec(n); cout<<endl<<"Factorial Result is:: "<<fact<<endl; getch(); } rec(int x) {

int f; if(x==1) return(x); else {

f=x\*rec(x-1); return(f); }

}

32. Program to compute the nth number of Fibonacci series using Recursion. Fibonacci series is given by 0, 1, 1, 2, 3, 5, 8, 13, 21, ---- #include<iostream.h> #include<conio.h> int recfib(int n) { if(n==1)

return 0; else if(n<3)

return 1; else return (recfib(n-1)+recfib(n-2));

} void main() { clrscr(); int n; cout<<"Fibonacci series generation using recursion"; cout<<"Enter the limit"; cin>>n; for(int i=1;i<=n;i++) cout<<recfib(i)<<" ";

getch(); }

33. Program to implement Tower of Hanoi problem #include<iostream.h> void move(int n,char \*s,char \*i,char \*d) { if(n>0) { move(n-1,s,d,i); cout<<"disk "<<n<<" is moved from "<<s<<" to "<<d<<endl; move(n-1,i,s,d); } } void main() { cout<<"Enter the no. of disks "; int n; cin>>n; move(n,"sourcetower","intermediatetower","destination tower"); }

34. Code to PUSH item in Stack using Linked List

push() { int value; struct node \*ptr; cout<<"\nPUSH Operationn"; cout<<"Enter a number to insert: "; cin>>value; ptr=new node; ptr->data=value; ptr->next=NULL; if(top!=NULL) ptr->next=top; top=ptr; cout<<"\nNew item is inserted to the stack!!!"; }

35. Code to POP item from stack using Linked List

pop() { struct node \*temp; if(top==NULL) { cout<<"\nThe stack is empty!!!"; } temp=top; top=top->next; cout<<"\nPOP Operation........nPoped value is "<<temp->data;

delete temp; }

36. Program to insert item in a Queue using Linked List #include<iostream> using namespace std; struct node { int data; node \*next; }\*front = NULL,\*rear = NULL,\*p = NULL,\*np = NULL; void push(int x) { np = new node; np->data = x; np->next = NULL; if(front == NULL) { front = rear = np; rear->next = NULL; } else { rear->next = np; rear = np; rear->next = NULL; } } int main() { int n,c = 0,x; cout<<"Enter the number of values to be pushed into queue"; cin>>n; while (c < n) { cout<<"Enter the value to be entered into queue"; cin>>x; push(x); c++; } }

37. Program to delete item from a Queue using Linked List #include<iostream> using namespace std; struct node { int data; node \*next; }\*front = NULL,\*rear = NULL,\*p = NULL,\*np = NULL;

int remove() { int x; if(front == NULL) { cout<<"empty queue"; } else { p = front; x = p->data; front = front->next; delete(p); return(x); } }

int main() { cout<<"Removed Values"; while(true) { if (front != NULL) cout<<remove()<<endl; else break; } return 0; }

38. Code to find element in the Binary Search Tree

struct node { int info; struct node \*left; struct node \*right; }\*root;

void BST::find(int item, node \*\*par, node \*\*loc) { node \*ptr, \*ptrsave; if (root == NULL) { \*loc = NULL; \*par = NULL; return; } if (item == root->info) { \*loc = root; \*par = NULL; return;

} if (item < root->info) ptr = root->left; else ptr = root->right; ptrsave = root; while (ptr != NULL) { if (item == ptr->info) { \*loc = ptr; \*par = ptrsave; return; } ptrsave = ptr; if (item < ptr->info) ptr = ptr->left; else ptr = ptr->right; } \*loc = NULL; \*par = ptrsave; }

39. Code to insert element into Binary Search Tree

struct node { int info; struct node \*left; struct node \*right; }\*root;

void BST::insert(node \*tree, node \*newnode) { if (root == NULL) { root = new node; root->info = newnode->info; root->left = NULL; root->right = NULL; cout<<"Root Node is Added"<<endl; return; } if (tree->info == newnode->info) { cout<<"Element already in the tree"<<endl; return; } if (tree->info > newnode->info) { if (tree->left != NULL) {

insert(tree->left, newnode); } else { tree->left = newnode; (tree->left)->left = NULL; (tree->left)->right = NULL; cout<<"Node Added To Left"<<endl; return; } } else { if (tree->right != NULL) { insert(tree->right, newnode); } else { tree->right = newnode; (tree->right)->left = NULL; (tree->right)->right = NULL; cout<<"Node Added To Right"<<endl; return; } } }

40. Code for Preorder Traversal of Binary Search Tree

void BST::preorder(node \*ptr) { if (root == NULL) { cout<<"Tree is empty"<<endl; return; } if (ptr != NULL) { cout<<ptr->info<<" "; preorder(ptr->left); preorder(ptr->right); } }

41. Code for Inorder Traversal of Binary Search Tree

void BST::inorder(node \*ptr) { if (root == NULL) { cout<<"Tree is empty"<<endl; return;

} if (ptr != NULL) { inorder(ptr->left); cout<<ptr->info<<" "; inorder(ptr->right); } }

42. Code for Postorder Traversal of Binary Search Tree

void BST::postorder(node \*ptr) { if (root == NULL) { cout<<"Tree is empty"<<endl; return; } if (ptr != NULL) { postorder(ptr->left); postorder(ptr->right); cout<<ptr->info<<" "; } }

43. Code for Displaying Binary Search Tree Structure

void BST::display(node \*ptr, int level) { int i; if (ptr != NULL) { display(ptr->right, level+1); cout<<endl; if (ptr == root) cout<<"Root->: "; else { for (i = 0;i < level;i++) cout<<" "; } cout<<ptr->info; display(ptr->left, level+1); } }

44. Code to create Circular Linked List

struct node { int info; struct node \*next; }\*last;

void circular\_llist::create\_node(int value) { struct node \*temp; temp = new(struct node); temp->info = value; if (last == NULL) { last = temp; temp->next = last; } else { temp->next = last->next; last->next = temp; last = temp; } }

45. Code to insert element in a Circular Linked List

struct node { int info; struct node \*next; }\*last;

void circular\_llist::add\_begin(int value) { if (last == NULL) { cout<<"First Create the list."<<endl; return; } struct node \*temp; temp = new(struct node); temp->info = value; temp->next = last->next; last->next = temp; }

46. Code to insert element at a particular place in a Circular Linked List

struct node { int info; struct node \*next; }\*last;

void circular\_llist::add\_after(int value, int pos) { if (last == NULL) { cout<<"First Create the list."<<endl; return; }

struct node \*temp, \*s; s = last->next; for (int i = 0;i < pos-1;i++) { s = s->next; if (s == last->next) { cout<<"There are less than "; cout<<pos<<" in the list"<<endl; return; } } temp = new(struct node); temp->next = s->next; temp->info = value; s->next = temp; /\*Element inserted at the end\*/ if (s == last) { last=temp; } }

47. Code to delete element from a Circular Linked List struct node { int info; struct node \*next; }\*last;

void circular\_llist::delete\_element(int value) { struct node \*temp, \*s; s = last->next; /\* If List has only one element\*/ if (last->next == last && last->info == value) { temp = last; last = NULL; free(temp); return; } if (s->info == value) /\*First Element Deletion\*/ { temp = s; last->next = s->next; free(temp); return; } while (s->next != last) { /\*Deletion of Element in between\*/

if (s->next->info == value) { temp = s->next; s->next = temp->next; free(temp); cout<<"Element "<<value; cout<<" deleted from the list"<<endl; return; } s = s->next; } /\*Deletion of last element\*/ if (s->next->info == value) { temp = s->next; s->next = last->next; free(temp); last = s; return; } cout<<"Element "<<value<<" not found in the list"<<endl; }

48. Code to search element in a Circular Linked List struct node { int info; struct node \*next; }\*last;

void circular\_llist::search\_element(int value) { struct node \*s; int counter = 0; s = last->next; while (s != last) { counter++; if (s->info == value) { cout<<"Element "<<value; cout<<" found at position "<<counter<<endl; return; } s = s->next; } if (s->info == value) { counter++; cout<<"Element "<<value; cout<<" found at position "<<counter<<endl; return;

} cout<<"Element "<<value<<" not found in the list"<<endl; }

49. Code to display Circular Linked List struct node { int info; struct node \*next; }\*last;

void circular\_llist::display\_list() { struct node \*s; if (last == NULL) { cout<<"List is empty, nothing to display"<<endl; return; } s = last->next; cout<<"Circular Link List: "<<endl; while (s != last) { cout<<s->info<<"->"; s = s->next; } cout<<s->info<<endl; }

50. Code to update Circular Linked List struct node { int info; struct node \*next; }\*last;

void circular\_llist::update() { int value, pos, i; if (last == NULL) { cout<<"List is empty, nothing to update"<<endl; return; } cout<<"Enter the node position to be updated: "; cin>>pos; cout<<"Enter the new value: "; cin>>value; struct node \*s; s = last->next; for (i = 0;i < pos - 1;i++) { if (s == last)

{ cout<<"There are less than "<<pos<<" elements."; cout<<endl; return; } s = s->next; } s->info = value; cout<<"Node Updated"<<endl; }

51. Code to sort Circular Linked List struct node { int info; struct node \*next; }\*last;

void circular\_llist::sort() { struct node \*s, \*ptr; int temp; if (last == NULL) { cout<<"List is empty, nothing to sort"<<endl; return; } s = last->next; while (s != last) { ptr = s->next; while (ptr != last->next) { if (ptr != last->next) { if (s->info > ptr->info) { temp = s->info; s->info = ptr->info; ptr->info = temp; } } else break; ptr = ptr->next; } s = s->next; } }

52. Code to create a node in a singly linked list

node \*single\_llist::create\_node(int value) { struct node \*temp, \*s; temp = new(struct node); if (temp == NULL) { cout<<"Memory not allocated "<<endl; return 0; } else { temp->info = value; temp->next = NULL; return temp; } }

53. Code to insert element at the beginning of a singly linked list

void single\_llist::insert\_begin() { int value; cout<<"Enter the value to be inserted: "; cin>>value; struct node \*temp, \*p; temp = create\_node(value); if (start == NULL) { start = temp; start->next = NULL; } else { p = start; start = temp; start->next = p; } cout<<"Element Inserted at beginning"<<endl; }

54. Code to insert element at the last position in a singly linked list

void single\_llist::insert\_last() { int value; cout<<"Enter the value to be inserted: "; cin>>value; struct node \*temp, \*s; temp = create\_node(value); s = start;

while (s->next != NULL) { s = s->next; } temp->next = NULL; s->next = temp; cout<<"Element Inserted at last"<<endl; }

55. Code to insert element at a given position in a singly linked list

void single\_llist::insert\_pos() { int value, pos, counter = 0; cout<<"Enter the value to be inserted: "; cin>>value; struct node \*temp, \*s, \*ptr; temp = create\_node(value); cout<<"Enter the postion at which node to be inserted: "; cin>>pos; int i; s = start; while (s != NULL) { s = s->next; counter++; } if (pos == 1) { if (start == NULL) { start = temp; start->next = NULL; } else { ptr = start; start = temp; start->next = ptr; } } else if (pos > 1 && pos <= counter) { s = start; for (i = 1; i < pos; i++) { ptr = s; s = s->next; } ptr->next = temp; temp->next = s;

} else { cout<<"Positon out of range"<<endl; } }

56. Code to sort a singly linked list

void single\_llist::sort() { struct node \*ptr, \*s; int value; if (start == NULL) { cout<<"The List is empty"<<endl; return; } ptr = start; while (ptr != NULL) { for (s = ptr->next;s !=NULL;s = s->next) { if (ptr->info > s->info) { value = ptr->info; ptr->info = s->info; s->info = value; } } ptr = ptr->next; } }

57. Code to delete node at a given position in a singly linked list

void single\_llist::delete\_pos() { int pos, i, counter = 0; if (start == NULL) { cout<<"List is empty"<<endl; return; } cout<<"Enter the position of value to be deleted: "; cin>>pos; struct node \*s, \*ptr; s = start; if (pos == 1) { start = s->next; }

else { while (s != NULL) { s = s->next; counter++; } if (pos > 0 && pos <= counter) { s = start; for (i = 1;i < pos;i++) { ptr = s; s = s->next; } ptr->next = s->next; } else { cout<<"Position out of range"<<endl; } free(s); cout<<"Element Deleted"<<endl; } }

58. Code to update a given node in a singly linked list

void single\_llist::update() { int value, pos, i; if (start == NULL) { cout<<"List is empty"<<endl; return; } cout<<"Enter the node position to be updated: "; cin>>pos; cout<<"Enter the new value: "; cin>>value; struct node \*s, \*ptr; s = start; if (pos == 1) start->info = value; else { for (i = 0;i < pos - 1;i++) { if (s == NULL) { cout<<"There are less than "<<pos<<" elements"; return;

} s = s->next; } s->info = value; } cout<<"Node Updated"<<endl; }

59. Code to search an element in a singly linked list

void single\_llist::search() { int value, pos = 0; bool flag = false; if (start == NULL) { cout<<"List is empty"<<endl; return; } cout<<"Enter the value to be searched: "; cin>>value; struct node \*s; s = start; while (s != NULL) { pos++; if (s->info == value) { flag = true; cout<<"Element "<<value<<" is found at position "<<pos<<endl; } s = s->next; } if (!flag) cout<<"Element "<<value<<" not found in the list"<<endl; }

60. Code to reverse a singly linked list

void single\_llist::reverse() { struct node \*ptr1, \*ptr2, \*ptr3; if (start == NULL) { cout<<"List is empty"<<endl; return; } if (start->next == NULL) return; ptr1 = start; ptr2 = ptr1->next; ptr3 = ptr2->next; ptr1->next = NULL;

ptr2->next = ptr1; while (ptr3 != NULL) { ptr1 = ptr2; ptr2 = ptr3; ptr3 = ptr3->next; ptr2->next = ptr1; } start = ptr2; }

61. Code to display the elements of singly linked list

void single\_llist::display() { struct node \*temp; if (start == NULL) { cout<<"The List is Empty"<<endl; return; } temp = start; cout<<"Elements of list are: "<<endl; while (temp != NULL) { cout<<temp->info<<"->"; temp = temp->next; } cout<<"NULL"<<endl; }

62. Code to insert an element in a circular queue

void insert(int item) { if ((front == 0 && rear == MAX-1) || (front == rear+1)) { cout<<"Queue Overflow \n"; return; } if (front == -1) { front = 0; rear = 0; } else { if (rear == MAX - 1) rear = 0; else rear = rear + 1; }

cqueue\_arr[rear] = item ; }

63. Code to delete an element from a circular queue

void del() { if (front == -1) { cout<<"Queue Underflow\n"; return ; } cout<<"Element deleted from queue is : "<<cqueue\_arr[front]<<endl; if (front == rear) { front = -1; rear = -1; } else { if (front == MAX - 1) front = 0; else front = front + 1; } }

64. Code to display Circular Queue

void display() { int front\_pos = front, rear\_pos = rear; if (front == -1) { cout<<"Queue is empty\n"; return; } cout<<"Queue elements :\n"; if (front\_pos <= rear\_pos) { while (front\_pos <= rear\_pos) { cout<<cqueue\_arr[front\_pos]<<" "; front\_pos++; } } else { while (front\_pos <= MAX - 1) { cout<<cqueue\_arr[front\_pos]<<" "; front\_pos++; }

front\_pos = 0; while (front\_pos <= rear\_pos) { cout<<cqueue\_arr[front\_pos]<<" "; front\_pos++; } } cout<<endl; }

65. Code to insert an element in a priority queue

void insert(int item, int priority) { node \*tmp, \*q; tmp = new node; tmp->info = item; tmp->priority = priority; if (front == NULL || priority < front->priority) { tmp->link = front; front = tmp; } else { q = front; while (q->link != NULL && q->link->priority <= priority) q=q->link; tmp->link = q->link; q->link = tmp; } }

66. Code to delete an element from priority queue

void del() { node \*tmp; if(front == NULL) cout<<"Queue Underflow\n"; else { tmp = front; cout<<"Deleted item is: "<<tmp->info<<endl; front = front->link; free(tmp); } }

67. Code to display priority queue

void display() { node \*ptr;

ptr = front; if (front == NULL) cout<<"Queue is empty\n"; else { cout<<"Queue is :\n"; cout<<"Priority Item\n"; while(ptr != NULL) { cout<<ptr->priority<<" "<<ptr->info<<endl; ptr = ptr->link; } } }

68. Program to Check Binary Tree is Binary Search Tree

struct node { int data; node\* left; node\* right; }; int isBSTUtil(node\* node, int min, int max); int isBST(node\* node) return(isBSTUtil(node, INT\_MIN, INT\_MAX)); int isBSTUtil(struct node\* node, int min, int max) { if (node==NULL) return 1; if (node->data < min || node->data > max) return 0; return isBSTUtil(node->left, min, node->data - 1) && isBSTUtil(node->right, node->data + 1, max); } node\* newNode(int data) { node\* nod = new node; nod->data = data; nod->left = NULL; nod->right = NULL; return nod; } int main() { node \*root = newNode(4); root->left = newNode(2); root->right = newNode(5); root->left->left = newNode(1); root->left->right = newNode(3); if (isBST(root)) cout<<"The Given Binary Tree is a BST"<<endl;

else cout<<"The Given Binary Tree is not a BST"<<endl; return 0; }

69. Code for Counting the total nodes in a tree

int countnode(treeptr root) { static int count=0; treeptr temp=root; if(temp!= NULL) { count ++; countnode(temp->left); countnode(temp->right); } return count; }

70. Code for Mirroring a given tree

void mirror(treeptr root) { treeptr temp=root, temp1; if(temp) { if(temp->left) mirror(temp->left); if(temp->right) mirror(temp->right); /\* interchange \*/ temp1=temp->left; temp->left=temp->right; temp->right=temp1; } }

71. Code for Comparing two binary search trees

int compare(treeptr root1, treeptr root2) { static int equal=0; if(root1==NULL && root2==NULL) return1; else if(root1!=NULL && root2!=NULL) if(root1 -> data == root2 -> data) if(compare(root1 -> left, root2 -> left)) equal = compare(root1 -> right,root2->right); else equal=0; return(equal); }

72. Code for Copying a tree

treeptr treecopy(treeptr root) { treeptr newnode; if(root!= NULL) { newnode=nodealloc;

newnode-> left = treecopy(root->left); newnode-> right = treecopy(root->right); newnode-> data = root->left; return(newnode); } else return NULL;

}

73. Program for Breadth First Search #include<iostream.h> #include<conio.h> #include<stdlib.h> int cost[10][10], i, j, k, n, queue[10], front, rear, v, visit[10], visited[10]; void main() { int m; clrscr(); cout <<"enter no of vertices"; cin >> n; cout <<"enter no of edges"; cin >> m; cout <<"\n EDGES \n"; for(k=1;k<=m;k++) { cin >>i>>j; cost[i][j]=1; } cout <<"enter initial vertex"; cin >>v; cout <<"Visited vertices\n"; cout << v; visited[v]=1; k=1; while(k<n)

{ for(j=1;j<=n;j++) if(cost[v][j]!=0 && visited[j]!=1 && visit[j]!=1) { visit[j]=1; queue[rear++]=j; } v=queue[front++]; cout<<v << " ";

k++; visit[v]=0; visited[v]=1; }

getch(); }

74. Program for Depth First Search #include<iostream.h> #include<conio.h> #include<stdlib.h> int cost[10][10],i,j,k,n,stack[10],top,v,visit[10],visited[10]; void main() { int m; cout <<"enter no of vertices"; cin >> n; cout <<"enter no of edges"; cin >> m; cout <<"\n EDGES \n";

for(k=1;k<=m;k++) { cin>>i>>j; cost[i][j]=1; } cout <<"enter initial vertex"; cin >>v; cout <<"ORDER OF VISITED VERTICES"; cout << v <<" "; visited[v]=1; k=1; while(k<n) { for(j=n;j>=1;j--) if(cost[v][j]!=0 && visited[j]!=1 && visit[j]!=1) { visit[j]=1; stack [top]=j; top++; } v= stack[--top]; cout<<v << " "; k++; visit[v]=0; visited[v]=1; } getch(); }

75. Code for search an element in Binary Threaded Tree

bool search(int key) { Node \*tmp = root->left;

for (;;) { if (tmp->key < key) { if (tmp->rightThread) return false; tmp = tmp->right; } else if (tmp->key > key) { if (tmp->leftThread) return false; tmp = tmp->left; } else { return true; } } }

76. Code to Print Binary Threaded Tree

void printTree() { Node \*tmp = root, \*p; for (;;) { p = tmp; tmp = tmp->right; if (!p->rightThread) { while (!tmp->leftThread) { tmp = tmp->left; } } if (tmp == root) break; cout<<tmp->key<<" "; } cout<<endl; }

77. Code to traverse B+ tree

void traverse(B+TreeNode \*p) { cout<<endl; int i; for (i = 0; i < p->n; i++) { if (p->leaf == false)

traverse(p->child\_ptr[i]); cout << " " << p->data[i]; } if (p->leaf == false) traverse(p->child\_ptr[i]); cout<<endl; }

78. Code to Sort B+ Tree

void sort(int \*p, int n) { int i, j, temp; for (i = 0; i < n; i++) { for (j = i; j <= n; j++) { if (p[i] > p[j]) { temp = p[i]; p[i] = p[j]; p[j] = temp; } } } }

79. Code to find the Height of AVL Tree

int avlTree::height(avl\_node \*temp) { int h = 0; if (temp != NULL) { int l\_height = height (temp->left); int r\_height = height (temp->right); int max\_height = max (l\_height, r\_height); h = max\_height + 1; } return h; }

80. Code to find Height Difference in AVL Tree

int avlTree::diff(avl\_node \*temp) { int l\_height = height (temp->left); int r\_height = height (temp->right); int b\_factor= l\_height - r\_height; return b\_factor; }

81. Code for balancing AVL Tree

avl\_node \*avlTree::balance(avl\_node \*temp) { int bal\_factor = diff (temp); if (bal\_factor > 1) { if (diff (temp->left) > 0) temp = ll\_rotation (temp); else temp = lr\_rotation (temp); } else if (bal\_factor < -1) { if (diff (temp->right) > 0) temp = rl\_rotation (temp); else temp = rr\_rotation (temp); } return temp; }

82. Code to Insert an element in AVL Tree

avl\_node \*avlTree::insert(avl\_node \*root, int value) { if (root == NULL) { root = new avl\_node; root->data = value; root->left = NULL; root->right = NULL; return root; } else if (value < root->data) { root->left = insert(root->left, value); root = balance (root); } else if (value >= root->data) { root->right = insert(root->right, value); root = balance (root); } return root; }

83. Code to display AVL Tree

void avlTree::display(avl\_node \*ptr, int level) { int i; if (ptr!=NULL) { display(ptr->right, level + 1);

printf("\n"); if (ptr == root) cout<<"Root -> "; for (i = 0; i < level && ptr != root; i++) cout<<" "; cout<<ptr->data; display(ptr->left, level + 1); } }

84. Code for InOrder Traversal of AVL Tree

void avlTree::inorder(avl\_node \*tree) { if (tree == NULL) return; inorder (tree->left); cout<<tree->data<<" "; inorder (tree->right); }

85. Code for PreOrder Traversal of AVL Tree

void avlTree::preorder(avl\_node \*tree) { if (tree == NULL) return; cout<<tree->data<<" "; preorder (tree->left); preorder (tree->right); }

86. Code for PostOrder Traversal of AVL Tree

void avlTree::postorder(avl\_node \*tree) { if (tree == NULL) return; postorder ( tree ->left ); postorder ( tree ->right ); cout<<tree->data<<" "; }

87. Code for Right-Right Rotation (RR) of AVL Tree

avl\_node \*avlTree::rr\_rotation(avl\_node \*parent) { avl\_node \*temp; temp = parent->right; parent->right = temp->left; temp->left = parent; return temp; }

88. Code for Left-Left Rotation (LL) of AVL Tree

avl\_node \*avlTree::ll\_rotation(avl\_node \*parent) { avl\_node \*temp; temp = parent->left; parent->left = temp->right; temp->right = parent; return temp; }

89. Code for Left-Right Rotation (LR) of AVL Tree

avl\_node \*avlTree::lr\_rotation(avl\_node \*parent) { avl\_node \*temp; temp = parent->left; parent->left = rr\_rotation (temp); return ll\_rotation (parent); }

90. Code for Right-Left Rotation (RL) of AVL Tree

avl\_node \*avlTree::rl\_rotation(avl\_node \*parent) { avl\_node \*temp; temp = parent->right; parent->right = ll\_rotation (temp); return rr\_rotation (parent); }

91. Program for the implementation of Breadth First Search (BFS) for a given graph

#include<iostream> #include<conio.h> #include<stdlib.h> using namespace std; int cost[10][10],i,j,k,n,qu[10],front,rare,v,visit[10],visited[10]; main() { int m; cout <<"enterno of vertices"; cin >> n; cout <<"ente no of edges"; cin >> m; cout <<"\nEDGES \n"; for(k=1;k<=m;k++) { cin >>i>>j; cost[i][j]=1; } cout <<"enter initial vertex";

cin >>v; cout <<"Visitied vertices\n"; cout << v; visited[v]=1; k=1; while(k<n) { for(j=1;j<=n;j++) if(cost[v][j]!=0 && visited[j]!=1 && visit[j]!=1) {

visit[j]=1; qu[rare++]=j; } v=qu[front++]; cout<<v << " "; k++; visit[v]=0; visited[v]=1;

} }

92. Program for the implementation of Depth-first search (DFS) for a given graph

#include<iostream> #include<conio.h> #include<stdlib.h> using namespace std; int cost[10][10],i,j,k,n,stk[10],top,v,visit[10],visited[10];

main() { int m; cout <<"enterno of vertices"; cin >> n; cout <<"ente no of edges"; cin >> m; cout <<"\nEDGES \n"; for(k=1;k<=m;k++) { cin >>i>>j; cost[i][j]=1; } cout <<"enter initial vertex"; cin >>v; cout <<"ORDER OF VISITED VERTICES"; cout << v <<" "; visited[v]=1; k=1; while(k<n) { for(j=n;j>=1;j--) if(cost[v][j]!=0 && visited[j]!=1 && visit[j]!=1){

visit[j]=1; stk[top]=j; top++; } v=stk[--top]; cout<<v << " "; k++; visit[v]=0; visited[v]=1; } }

93. Code to get Transpose of a Graph

Graph Graph::getTranspose() { Graph g(V); for (int v = 0; v < V; v++) { list<int>::iterator i; for(i = adj[v].begin(); i != adj[v].end(); ++i) { g.adj[\*i].push\_back(v); } } return g; }

94. Code for Deletion of element from the Doubly Linked List void double\_llist::delete\_element(int value) { struct node \*tmp, \*q; /\*first element deletion\*/ if (start->info == value) { tmp = start; start = start->next; start->prev = NULL; cout<<"Element Deleted"<<endl; free(tmp); return; } q = start; while (q->next->next != NULL) { /\*Element deleted in between\*/ if (q->next->info == value) { tmp = q->next; q->next = tmp->next; tmp->next->prev = q; cout<<"Element Deleted"<<endl; free(tmp);

return; } q = q->next; } /\*last element deleted\*/ if (q->next->info == value) { tmp = q->next; free(tmp); q->next = NULL; cout<<"Element Deleted"<<endl; return; } cout<<"Element "<<value<<" not found"<<endl; }

95. Code to insert at a particular position in a Doubly Linked List void double\_llist::add\_after(int value, int pos) { if (start == NULL) { cout<<"First Create the list."<<endl; return; } struct node \*tmp, \*q; int i; q = start; for (i = 0;i < pos - 1;i++) { q = q->next; if (q == NULL) { cout<<"There are less than "; cout<<pos<<" elements."<<endl; return; } } tmp = new(struct node); tmp->info = value; if (q->next == NULL) { q->next = tmp; tmp->next = NULL; tmp->prev = q; } else { tmp->next = q->next; tmp->next->prev = tmp; q->next = tmp;

tmp->prev = q; } cout<<"Element Inserted"<<endl; }

96. Program to find MST(Minimal Spanning Tree) using Prim's Algorithm #include <iostream> #include <conio.h> using namespace std; struct node { int fr, to, cost; }p[6]; int c = 0, temp1 = 0, temp = 0; void prims(int \*a, int b[][7], int i, int j) { a[i] = 1; while (c < 6) { int min = 999; for (int i = 0; i < 7; i++) { if (a[i] == 1) { for (int j = 0; j < 7; ) { if (b[i][j] >= min || b[i][j] == 0) j++; else if (b[i][j] < min) { min = b[i][j]; temp = i; temp1 = j; } } } } a[temp1] = 1; p[c].fr = temp; p[c].to = temp1; p[c].cost = min; c++; b[temp][temp1] = b[temp1][temp]=1000; } for (int k = 0; k < 6; k++) { cout<<"source node:"<<p[k].fr<<endl; cout<<"destination node:"<<p[k].to<<endl;

cout<<"weight of node"<<p[k].cost<<endl; } } int main() { int a[7]; for (int i = 0; i < 7; i++) a[i] = 0; int b[7][7]; for (int i = 0; i < 7; i++) { cout<<"enter values for "<<(i+1)<<" row"<<endl; for (int j = 0; j < 7; j++) cin>>b[i][j]; } prims(a, b, 0, 0); getch(); }

97. Program to Implement Floyd-Warshall Algorithm #include <iostream> #include <conio.h> using namespace std; void floyds(int b[][7]) { int i, j, k; for (k = 0; k < 7; k++) { for (i = 0; i < 7; i++) { for (j = 0; j < 7; j++) { if ((b[i][k] \* b[k][j] != 0) && (i != j)) { if ((b[i][k] + b[k][j] < b[i][j]) || (b[i][j] == 0)) b[i][j] = b[i][k] + b[k][j]; } } } } for (i = 0; i < 7; i++) { cout<<"\nMinimum Cost With Respect to Node:"<<i<<endl; for (j = 0; j < 7; j++) cout<<b[i][j]<<"\t"; } } int main() { int b[7][7]; cout<<"ENTER VALUES OF ADJACENCY MATRIX\n\n";

for (int i = 0; i < 7; i++) { cout<<"enter values for "<<(i+1)<<" row"<<endl; for (int j = 0; j < 7; j++) cin>>b[i][j]; } floyds(b); getch(); }

98. Code for graph using Adjacency List

void addReverseEdge(int src,int dest) { np1 = new adj\_list; np1->dest = src; np1->next = NULL; if (array[dest].ptr == NULL) { array[dest].ptr = np1; q = array[dest].ptr; q->next = NULL; } else { q = array[dest].ptr; while (q->next != NULL) { q = q->next; q->next = np1; } } void addEdge(int src,int dest) { np<!--more--> = new adj\_list; np->dest = dest; np->next = NULL; if (array[src].ptr == NULL) { array[src].ptr = np; p = array[src].ptr; p->next = NULL; } else { p = array[src].ptr; while (p->next != NULL) p = p->next; p->next = np; } addReverseEdge(src,dest); }

99. Program to Implement Adjacency Matrix #include <iostream> #include <cstdlib> using namespace std; #define MAX 20 class AdjacencyMatrix { private: int n; int \*\*adj; bool \*visited; public: AdjacencyMatrix(int n) { this->n = n; visited = new bool [n]; adj = new int\* [n]; for (int i = 0; i < n; i++) { adj[i] = new int [n]; for(int j = 0; j < n; j++) adj[i][j] = 0; } } void add\_edge(int origin, int destin) { if( origin > n || destin > n || origin < 0 || destin < 0) cout<<"Invalid edge!\n"; else adj[origin - 1][destin - 1] = 1; } void display() { int i,j; for(i = 0;i < n;i++) { for(j = 0; j < n; j++) cout<<adj[i][j]<<" "; cout<<endl; } } }; int main() { int nodes, max\_edges, origin, destin; cout<<"Enter number of nodes: "; cin>>nodes; AdjacencyMatrix am(nodes); max\_edges = nodes \* (nodes - 1); for (int i = 0; i < max\_edges; i++) {

cout<<"Enter edge (-1 -1 to exit): "; cin>>origin>>destin; if((origin == -1) && (destin == -1)) break; am.add\_edge(origin, destin); } am.display(); return 0; }

100. Code for insertion at the beginning in the Doubly Linked List

void double\_llist::add\_begin(int value) { if (start == NULL) { cout<<"First Create the list."<<endl; return; } struct node \*temp; temp = new(struct node); temp->prev = NULL; temp->info = value; temp->next = start; start->prev = temp; start = temp; cout<<"Element Inserted"<<endl; }