Practical 1

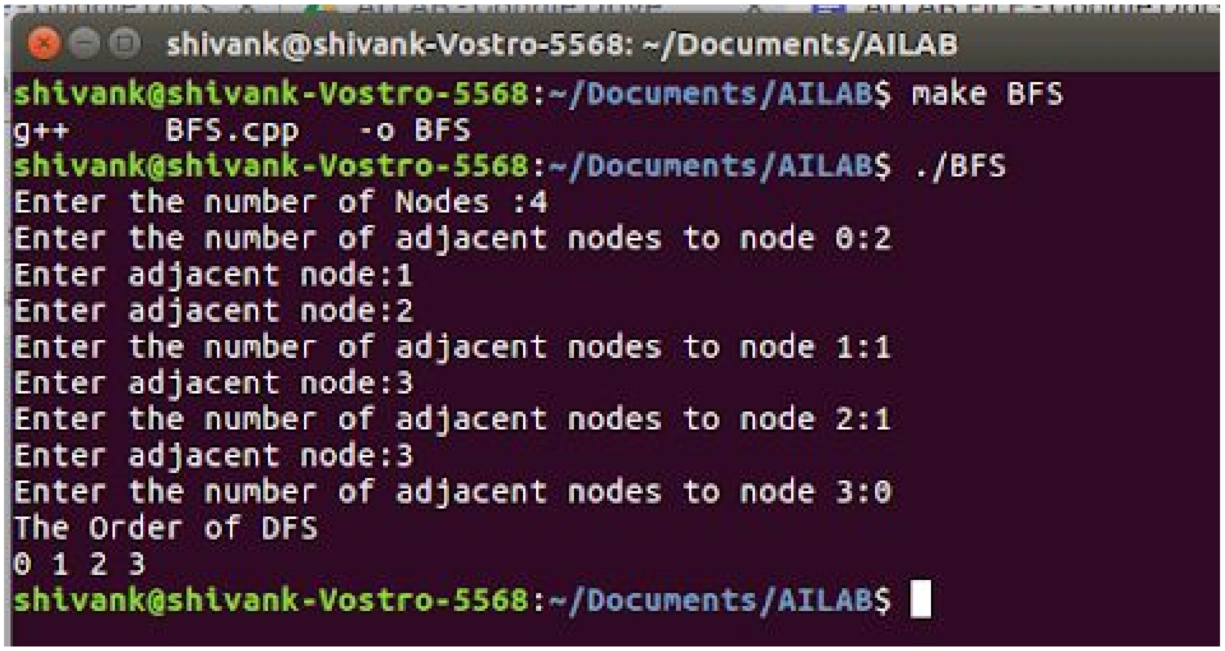
AIM : To implement BFS (Breadth First Search)

CODE

```
#include<iostream>
#include<vector>
using namespace std;

vector <int> s;
void EnterQ(int n){
        s.insert(s.end()-1,n);
}

int ExitQ(){
        int j=s.front();
        s.erase(s.begin());
        return j;
}

void BFS(int **G, int n){
        s.push_back(-1);
        EnterQ(0);
        vector <int> visited(n);
        visited[0]=1;
        int x=ExitQ();
        while(x!=-1){
                for(int i=1;i<=G[x][0];i++){
                        if(visited[G[x][i]]==0)
                        {
                                EnterQ(G[x][i]);
                                visited[G[x][i]]=1;
                        }
                }
                cout<<x<<' ';
                x=ExitQ();
        }
        cout<<endl;
}
```

```
int main(){
	int n;//number of Nodes;
	cout<<"Enter the number of Nodes :";
	cin>>n;

	int **G = new int*[n];

	for(int i=0,k;i<n;i++)
	{
		cout<<"Enter the number of adjacent nodes to node "<<i<<':';
		cin>>k;
		G[i] = new int[k+1];
		G[i][0]=k;
		for(int j=1;j<=k;j++){
			cout<<"Enter adjacent node:";
			cin>>G[i][j];
		}
	}
	cout<<"The Order of DFS\n";
	BFS(G,n);
	return 0;
}
```

OUTPUT



```
shivank@shivank-Vostro-5568: ~/Documents/AILAB
shivank@shivank-Vostro-5568:~/Documents/AILAB$ make BFS
g++     BFS.cpp   -o BFS
shivank@shivank-Vostro-5568:~/Documents/AILAB$ ./BFS
Enter the number of Nodes :4
Enter the number of adjacent nodes to node 0:2
Enter adjacent node:1
Enter adjacent node:2
Enter the number of adjacent nodes to node 1:1
Enter adjacent node:3
Enter the number of adjacent nodes to node 2:1
Enter adjacent node:3
Enter the number of adjacent nodes to node 3:0
The Order of DFS
0 1 2 3
shivank@shivank-Vostro-5568:~/Documents/AILAB$ 
```

Practical 2

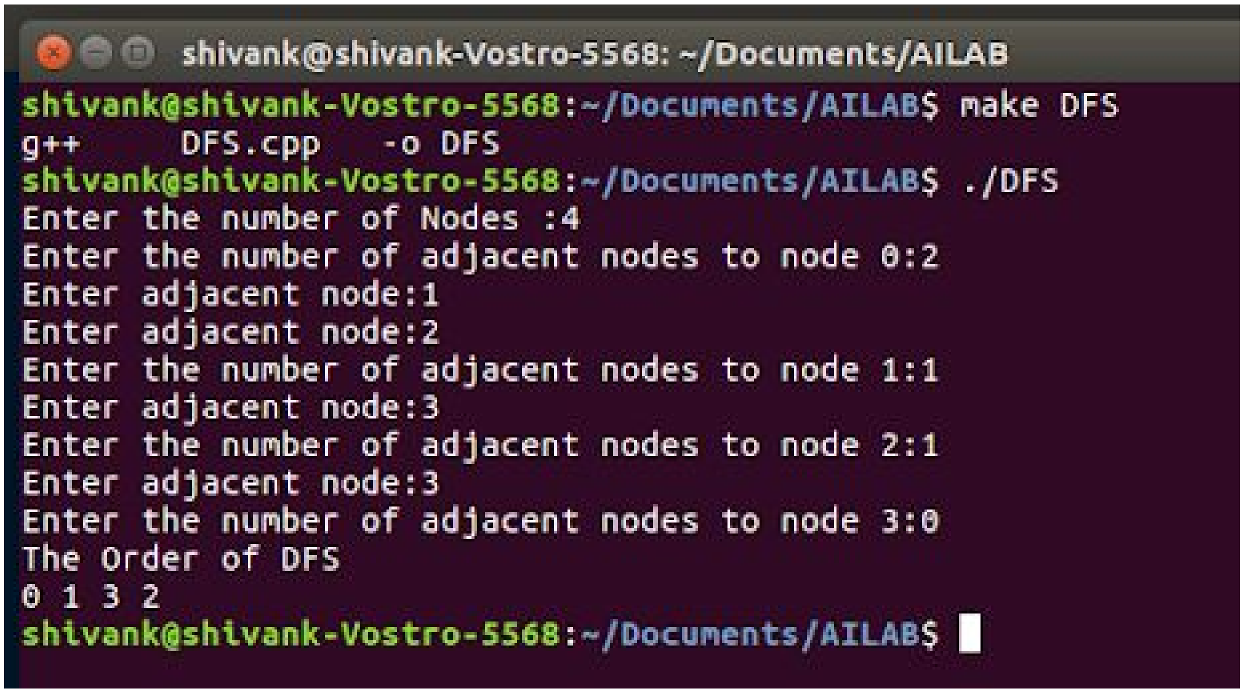
AIM : To implement DFS (Depth First Search)

CODE

```
#include<iostream>
#include<vector>
using namespace std;
vector <int> s;

void push(int n){
	s.push_back(n);
}

int pop(){
	int j=s.back();
	s.pop_back();
	return j;
}

void DFS(int **G, int n){
	push(-1);
	push(0);
	vector <int> visited(n);
	visited[0]=1;
	int x=pop();
	while(x!=-1){
		for(int i=G[x][0];i>0;i--){
			if(visited[G[x][i]]==0)
			{
				push(G[x][i]);
				visited[G[x][i]]=1;
			}
		}
		cout<<x<<' ';
		x=pop();
	}
	cout<<endl;
}
```

```
int main(){
        int n;//number of Nodes;
        cout<<"Enter the number of Nodes :";
        cin>>n;

        int **G = new int*[n];

        for(int i=0,k;i<n;i++)
        {
                cout<<"Enter the number of adjacent nodes to node "<<i<<':';
                cin>>k;
                G[i] = new int[k+1];
                G[i][0]=k;
                for(int j=1clear;j<=k;j++){
                        cout<<"Enter adjacent node:";
                        cin>>G[i][j];
                }
        }
        cout<<"The Order of DFS\n";
        DFS(G,n);
        return 0;
}
```

OUTPUT



shivank@shivank-Vostro-5568: ~/Documents/AILAB

```
shivank@shivank-Vostro-5568:~/Documents/AILAB$ make DFS
g++     DFS.cpp   -o DFS
shivank@shivank-Vostro-5568:~/Documents/AILAB$ ./DFS
Enter the number of Nodes :4
Enter the number of adjacent nodes to node 0:2
Enter adjacent node:1
Enter adjacent node:2
Enter the number of adjacent nodes to node 1:1
Enter adjacent node:3
Enter the number of adjacent nodes to node 2:1
Enter adjacent node:3
Enter the number of adjacent nodes to node 3:0
The Order of DFS
0 1 3 2
shivank@shivank-Vostro-5568:~/Documents/AILAB$
```

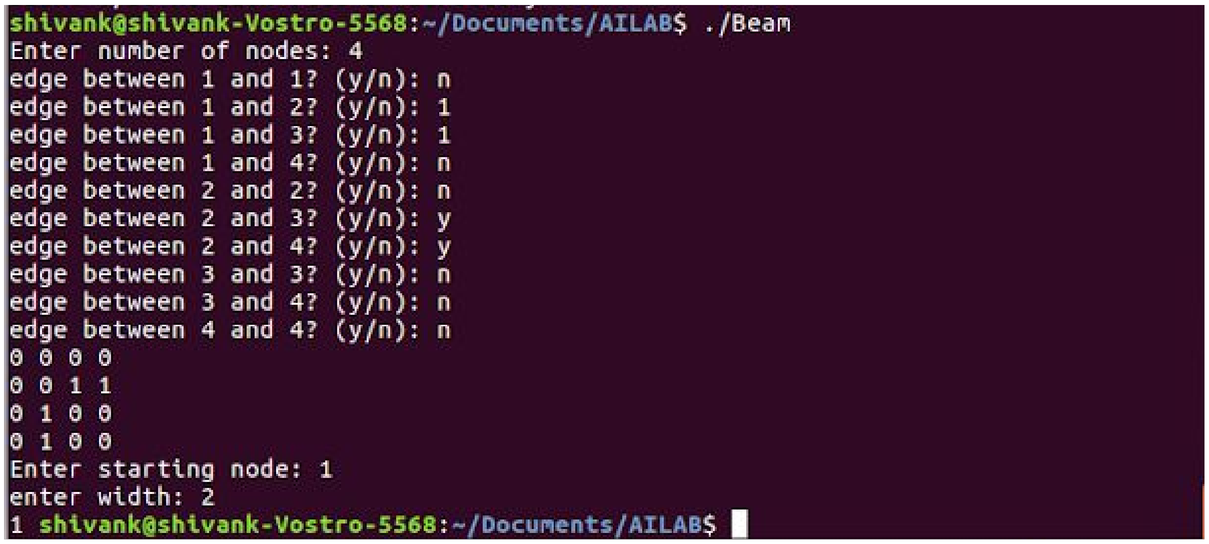
Practical 3

AIM : To implement Beam Search

CODE

```
#include<iostream>
using namespace std;
int queue_(int *,int ,int, int);
int dequeue_(int *, int , int, int );

int main()
{
   int n,i,j;
   cout<<"Enter number of nodes: ";
   cin>>n;
   int rowCount = n;
   int colCount = n ;
   int** edge = new int*[rowCount];
   for( i = 0; i < rowCount; ++i)
   {
      edge[i] = new int[colCount];
   }

   //edge adjacency matrix
   string ans;
   for(i = 0; i < n ; i++)
   {
      for(j = i ; j < n ; j++)
      {
         cout<<"edge between "<<i+1<< " and "<< j+1<<"? (y/n): ";
         cin>>ans;
         if(ans == "y")
         {
            edge[i][j] = 1;
            edge[j][i] = 1;
         }
         else
         {
            edge[i][j] = 0;
            edge[j][i] = 0;
         }
      }
   }
   //print edge adjacency matrix
   for(i = 0 ; i<n ; i++)
   {
      for(j=0; j<n;j++)
      {
         cout<<edge[i][j]<<" ";
      }
      cout<<endl;
   }

   //allocate memory to queue
   int *q = new int[n];
   int *flag = new int[n];
   for(i=0;i<n;i++)
   {
      q[i] = -1;
      flag[i] = 0;
   }

   int front_ = 0;
   int rear = -1;
   int start;
   cout<<"Enter starting node: ";
   cin>>start;
   int width;
   cout<<"enter width: ";
   cin>>width;
   start = start - 1;
   rear = queue_(q,start,n,rear);
   //cout<<"\nfront is "<<front_;
   //cout<<"\nrear is "<<rear;
   flag[start] = 1;
   while(front_ <= rear)
   {
        front_ = dequeue_(q,n,front_, rear);
        //cout<<"after dequeue\n";
        //cout<<"front is "<<front_<<"rear is "<<rear;
        cout<<start+1<<" ";
        int countt = 0;
        for(i = 0; i<n ; i ++)
        {
            if(edge[start][i] == 1 && flag[i] == 0 && countt<width)
            {
                countt++;
                rear = queue_(q,i,n,rear);
                flag[i]= 1;
            }
        }
        start = q[front_];
    }

}

int queue_(int *q, int start, int n, int rear)
{
    rear++;
    if(rear==n)
    {
        cout<<"overflow";
    }
    else
    {
        q[rear] = start;
    }
    return rear;

}

int dequeue_(int *q,int n,int front_, int rear)
{
    if(front_>rear)
    {
        cout<<"underflow";
    }
    else
    {
        q[front_] = -1;
        front_++;
    }
    return front_;
}
```

OUTPUT



```
shivank@shivank-Vostro-5568:~/Documents/AILAB$ ./Beam
Enter number of nodes: 4
edge between 1 and 1? (y/n): n
edge between 1 and 2? (y/n): 1
edge between 1 and 3? (y/n): 1
edge between 1 and 4? (y/n): n
edge between 2 and 2? (y/n): n
edge between 2 and 3? (y/n): y
edge between 2 and 4? (y/n): y
edge between 3 and 3? (y/n): n
edge between 3 and 4? (y/n): n
edge between 4 and 4? (y/n): n
0 0 0 0
0 0 1 1
0 1 0 0
0 1 0 0
Enter starting node: 1
enter width: 2
1 shivank@shivank-Vostro-5568:~/Documents/AILAB$
```

Practical 4

AIM : To implement Iterative Deepening Search (DFID Depth First Iterative Deepening search)

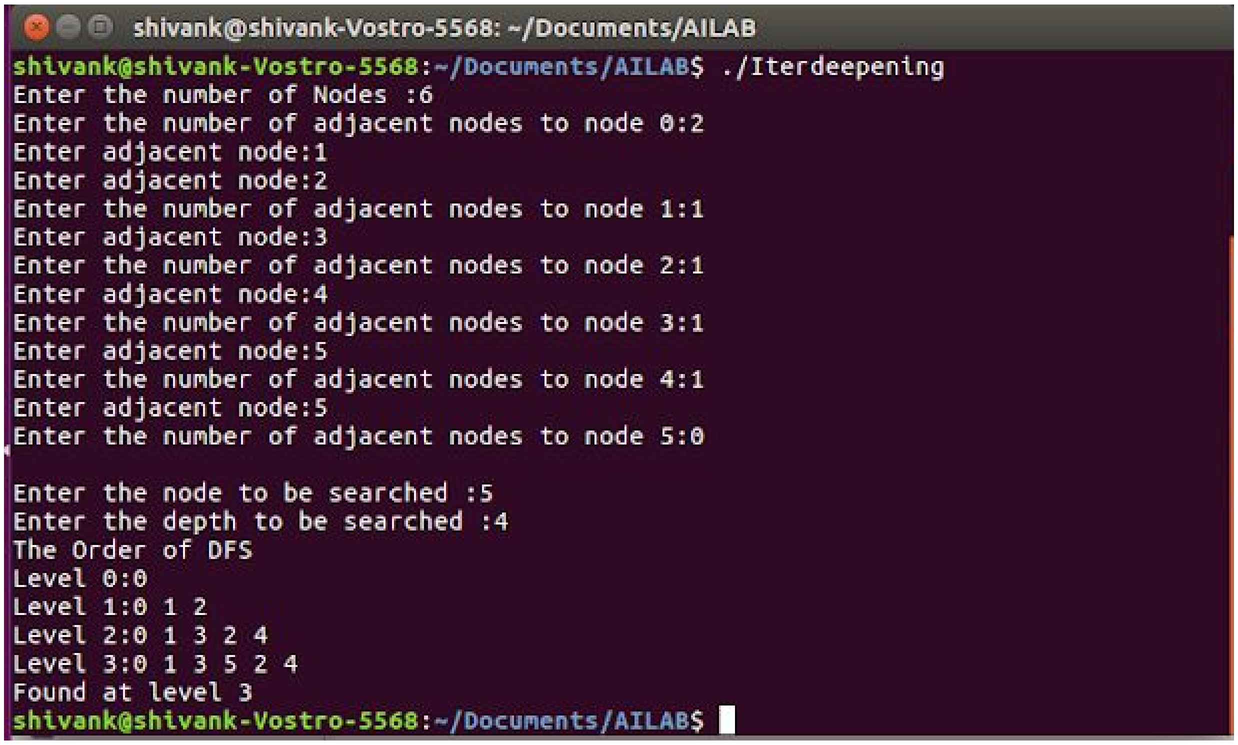
CODE

```
#include<iostream>
#include<vector>
using namespace std;

vector <int> s;

void push(int n){
        s.push_back(n);
}

int pop(){
        int j=s.back();
        s.pop_back();
        return j;
}


void IDS(int **G, int n, int t,int max_depth){
        cout<<"Level 0:"<<0;
        if(t==0){
                cout<<endl<<"Found at level 0 ";
                return;
        }
        int fl=0;
        for(int k=1;k<=max_depth;k++){
                int q;
                cout<<endl<<"Level "<<k<<":";
                s.clear();
                s.push_back(-1);
                vector <int> visited(n);
                for(int i=0;i<n;i++)visited[i]=-1;
                visited[0]=0;
                int x=0;
```

```
		while(x!=-1){
			if(x==t)fl=1;
			for(int i=G[x][0];i>0;i--){
				if(visited[G[x][i]]==-1)
				{
					if(k-(visited[x]+1)>=0)
					{
						push(G[x][i]);
						visited[G[x][i]]=visited[x]+1;
					}
				}
			}
			cout<<x<<' ';
			x=pop();

		}
		if(fl==1){
			cout<<endl<<"Found at level "<<k<<endl;
			break;
		}
	}
	if(fl==0){
		cout<<endl<<"Element not found";
	}
}

int main(){
	int n,x,d;//number of Nodes;
	cout<<"Enter the number of Nodes :";
	cin>>n;

	int **G = new int*[n];

	for(int i=0,k;i<n;i++)
	{
		cout<<"Enter the number of adjacent nodes to node "<<i<<':';
		cin>>k;
		G[i] = new int[k+1];
		G[i][0]=k;
```

```
                for(int j=1;j<=k;j++){
                        cout<<"Enter adjacent node:";
                        cin>>G[i][j];
                }
        }

        cout<<endl;
        cout<<"Enter the node to be searched :";
        cin>>x;
        cout<<"Enter the depth to be searched :";
        cin>>d;
        cout<<"The Order of DFS\n";
        IDS(G,n,x,d);
        return 0;
}
```

OUTPUT



```
shivank@shivank-Vostro-5568: ~/Documents/AILAB
shivank@shivank-Vostro-5568:~/Documents/AILAB$ ./Iterdeepening
Enter the number of Nodes :6
Enter the number of adjacent nodes to node 0:2
Enter adjacent node:1
Enter adjacent node:2
Enter the number of adjacent nodes to node 1:1
Enter adjacent node:3
Enter the number of adjacent nodes to node 2:1
Enter adjacent node:4
Enter the number of adjacent nodes to node 3:1
Enter adjacent node:5
Enter the number of adjacent nodes to node 4:1
Enter adjacent node:5
Enter the number of adjacent nodes to node 5:0

Enter the node to be searched :5
Enter the depth to be searched :4
The Order of DFS
Level 0:0
Level 1:0 1 2
Level 2:0 1 3 2 4
Level 3:0 1 3 5 2 4
Found at level 3
shivank@shivank-Vostro-5568:~/Documents/AILAB$
```

Practical 5

AIM : Program to implement Tic Tac Toe game.

CODE

```
#include<bits/stdc++.h>
using namespace std;

char board[3][3]={      {'-', '-', '-'},
                        {'-', '-', '-'},
                        {'-', '-', '-'} };

char human='X',ai='O',blank='-';
int row,clm;
char draw='d';
char ndraw='n';

void Hmove()
{
        cout<<"Enter the row number and column number {eg.:- 2 1} :";
        cin>>row>>clm;
        board[row-1][clm-1]=human;
}


int Cmove()
{
        if(board[1][1]==blank)
        {
                board[1][1]=ai;
        }
        else
        {
                //if ai is winning in this move
                for(int i=0;i<3;i++)
                {
                        //row
                        if(board[i][0]==board[i][1] && board[i][1]==ai && board[i][2]==blank)
                        {
```

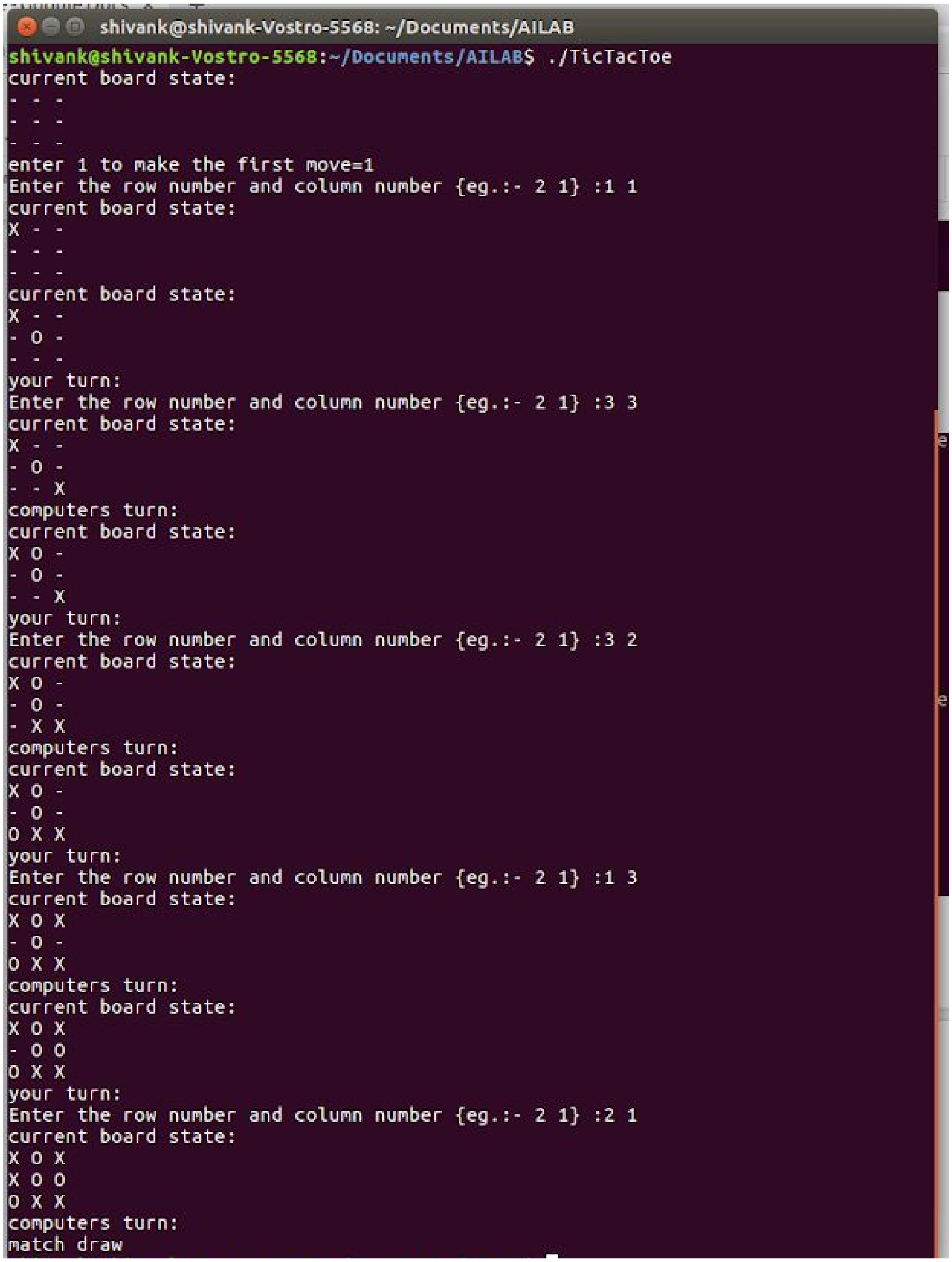
```
				board[i][2]=ai;
				return 1;
			}
			else if(board[i][1]==board[i][2] && board[i][2]==ai &&
board[i][0]==blank)
			{
				board[i][0]=ai;
				return 1;
			}
			else if(board[i][0]==board[i][2] && board[i][2]==ai &&
board[i][1]==blank)
			{
				board[i][1]=ai;
				return 1;
			}

			//column
			if(board[0][i]==board[1][i] && board[1][i]==ai && board[2][i]==blank)
			{
				board[2][i]=ai;
				return 1;
			}
			else if(board[1][i]==board[2][i] && board[2][i]==ai &&
board[0][i]==blank)
			{
				board[0][i]=ai;
				return 1;
			}
			else if(board[0][i]==board[2][i] && board[2][i]==ai &&
board[1][i]==blank)
			{
				board[1][i]=ai;
				return 1;
			}

			//diagonal
			if(board[0][0]==board[1][1] && board[1][1]==ai &&
board[2][2]==blank)
			{
				board[2][2]=ai;
				return 1;
			}
			else if(board[1][1]==board[2][2] && board[2][2]==ai &&
board[0][0]==blank)
			{
				board[0][0]=ai;
				return 1;
			}
			else if(board[0][0]==board[2][2] && board[2][2]==ai &&
board[1][1]==blank)
			{
				board[1][1]=ai;
				return 1;
			}
			else if(board[0][2]==board[1][1] && board[1][1]==ai &&
board[2][0]==blank)
			{
				board[2][0]=ai;
				return 1;
			}
			else if(board[1][1]==board[2][0] && board[2][0]==ai &&
board[0][2]==blank)
			{
				board[0][2]=ai;
				return 1;
			}
			else if(board[0][2]==board[2][0] && board[2][0]==ai &&
board[1][1]==blank)
			{
				board[1][1]=ai;
				return 1;
			}
		}

		//if human is winning in next move
		for(int i=0;i<3;i++)
		{
			//row
			if(board[i][0]==board[i][1] && board[i][0]==human &&
board[i][2]==blank)
			{
				board[i][2]=ai;
				return 1;

			}
			else if(board[i][1]==board[i][2] && board[i][2]==human &&
board[i][0]==blank)
			{
				board[i][0]=ai;
				return 1;
			}
			else if(board[i][0]==board[i][2] && board[i][2]==human &&
board[i][1]==blank)
			{
				board[i][1]=ai;
				return 1;
			}

			//column
			if(board[0][i]==board[1][i] && board[1][i] && board[1][i]==human &&
board[2][i]==blank)
			{
				board[2][i]=ai;
				return 1;
			}
			else if(board[1][i]==board[2][i] && board[2][i]==human &&
board[0][i]==blank)
			{
				board[0][i]=ai;
				return 1;
			}
			else if(board[0][i]==board[2][i] && board[2][i]==human &&
board[1][i]==blank)
			{
				board[1][i]=ai;
				return 1;
			}
			//diagonal
			if(board[0][0]==board[1][1] && board[1][1]==human &&
board[2][2]==blank)
			{
				board[2][2]=ai;
				return 1;
			}
			else if(board[1][1]==board[2][2] && board[2][2]==human &&
board[0][0]==blank)
			{
				board[0][0]=ai;
				return 1;
			}
			else if(board[0][0]==board[2][2] && board[2][2]==human &&
board[1][1]==blank)
			{
				board[1][1]=ai;
				return 1;
			}
			else if(board[0][2]==board[1][1] && board[1][1]==human &&
board[2][0]==blank)
			{
				board[2][0]=ai;
				return 1;
			}
			else if(board[1][1]==board[2][0] && board[2][0]==human &&
board[0][2]==blank)
			{
				board[0][2]=ai;
				return 1;
			}
			else if(board[0][2]==board[2][0] && board[2][0]==human &&
board[1][1]==blank)
			{
				board[1][1]=ai;
				return 1;
			}
		}
		//if neither human wins in next move nor ai wins in this move
		if(board[0][1]==blank)
		{
			board[0][1]=ai;
			return 1;
		}
		else if(board[1][0]==blank)
		{
			board[1][0]=ai;
			return 1;
		}
		else if(board[1][2]==blank)
		{
			board[1][2]=ai;
			return 1;
		}
		else
		{
			board[2][1]=ai;
			return 1;
		}
	}
	return 1;
}


void display()
{
	cout<<"current board state:"<<endl;
	for(int i=0;i<3;i++)
	{
		for(int j=0;j<3;j++)
		{
			cout<<board[i][j]<<" ";
		}
		cout<<endl;
	}
}
```

```
int isEmpty()
{
	for(int i=0;i<3;i++)
	{
		for(int j=0;j<3;j++)
		{
			if(board[i][j]==blank)
				return 1;
		}
	}
	return 0;
}


int isWin()
{
	char temp;
	for(int i=0;i<3;i++)
	{
		//row
		if(board[i][0]==board[i][1] && board[i][1]==board[i][2] &&
board[i][0]==board[i][2])
		{
			if(board[i][0]=='x')
				temp = 'x';
			else if(board[i][0]=='o')
				temp = 'o';
		}

		//column
		if(board[0][i]==board[1][i] && board[1][i]==board[2][i] &&
board[0][i]==board[2][i])
		{
			if(board[0][i]=='x')
				temp = 'x';
			else if(board[0][i]=='o')
				temp = 'o';
		}

		//diagonals
		if(board[0][0]==board[1][1] && board[1][1]==board[2][2] &&
board[0][0]==board[2][2])
		{
			if(board[0][0]=='x')
				temp = 'x';
			else if(board[0][0]=='o')
				temp = 'o';
		}

		if(board[0][2]==board[1][1] && board[1][1]==board[2][0] &&
board[0][2]==board[2][0])
		{
			if(board[0][2]=='x')
				temp = 'x';
			else if(board[0][2]=='o')
				temp = 'o';
		}
	}
	return temp;

	//check for draw
	/*int chk=isEmpty();
	if(chk==1)
		return ndraw;
	else
		return draw;*/
}


int main()
{
	display();
	cout<<"enter 1 to make the first move=";
	int move;
	cin>>move;
	while(1)
```

```
{
	if(move==1)
	{
		Hmove();
		display();
		move++;
	}
	else
	{
		int temp=isEmpty(); //returns  1 if any one is blank
		                                        // else it is checkked for result
		if(temp==1)
		{
			Cmove(); // computers move
			char win=isWin();//checks if anyone has one !
			display();
			if(win==human)
			{
				cout<<"congratulation: you win"<<endl;
				break;
			}
			else if(win==ai)
			{
				cout<<"Try again: you lose"<<endl;
				break;
			}
			else if(win==draw)
			{
				cout<<"match draw"<<endl;
				break;
			}
			else
				cout<<"your turn:"<<endl;
			Hmove();
			display();
			win=isWin();
			if(win==human)
			{
				cout<<"congratulation: you win"<<endl;
```

```
                                        break;
                                }
                                else if(win==ai)
                                {
                                        cout<<"Try again: you lose"<<endl;
                                        break;
                                }
                                else if(win==draw)
                                {
                                        cout<<"match draw"<<endl;
                                        break;
                                }
                                else
                                        cout<<"computers turn:"<<endl;
                        }
                        else
                        {
                                char win=isWin();
                                if(win==human)
                                {
                                        cout<<"congratulation: you win"<<endl;
                                        break;
                                }
                                else if(win==ai)
                                {
                                        cout<<"Try again: you lose"<<endl;
                                        break;
                                }
                                else
                                {
                                        cout<<"match draw"<<endl;
                                        break;
                                }
                        }
                }
        }
        return 0;
}
```

OUTPUT



shivank@shivank-Vostro-5568: ~/Documents/AILAB

```
shivank@shivank-Vostro-5568:~/Documents/AILAB$ ./TicTacToe
current board state:
- - -
- - -
- - -
enter 1 to make the first move=1
Enter the row number and column number {eg.:- 2 1} :1 1
current board state:
X - -
- - -
- - -
current board state:
X - -
- O -
- - -
your turn:
Enter the row number and column number {eg.:- 2 1} :3 3
current board state:
X - -
- O -
- - X
computers turn:
current board state:
X O -
- O -
- - X
your turn:
Enter the row number and column number {eg.:- 2 1} :3 2
current board state:
X O -
- O -
- X X
computers turn:
current board state:
X O -
- O -
O X X
your turn:
Enter the row number and column number {eg.:- 2 1} :1 3
current board state:
X O X
- O -
O X X
computers turn:
current board state:
X O X
- O O
O X X
your turn:
Enter the row number and column number {eg.:- 2 1} :2 1
current board state:
X O X
X O O
O X X
computers turn:
match draw
```

Practical 6

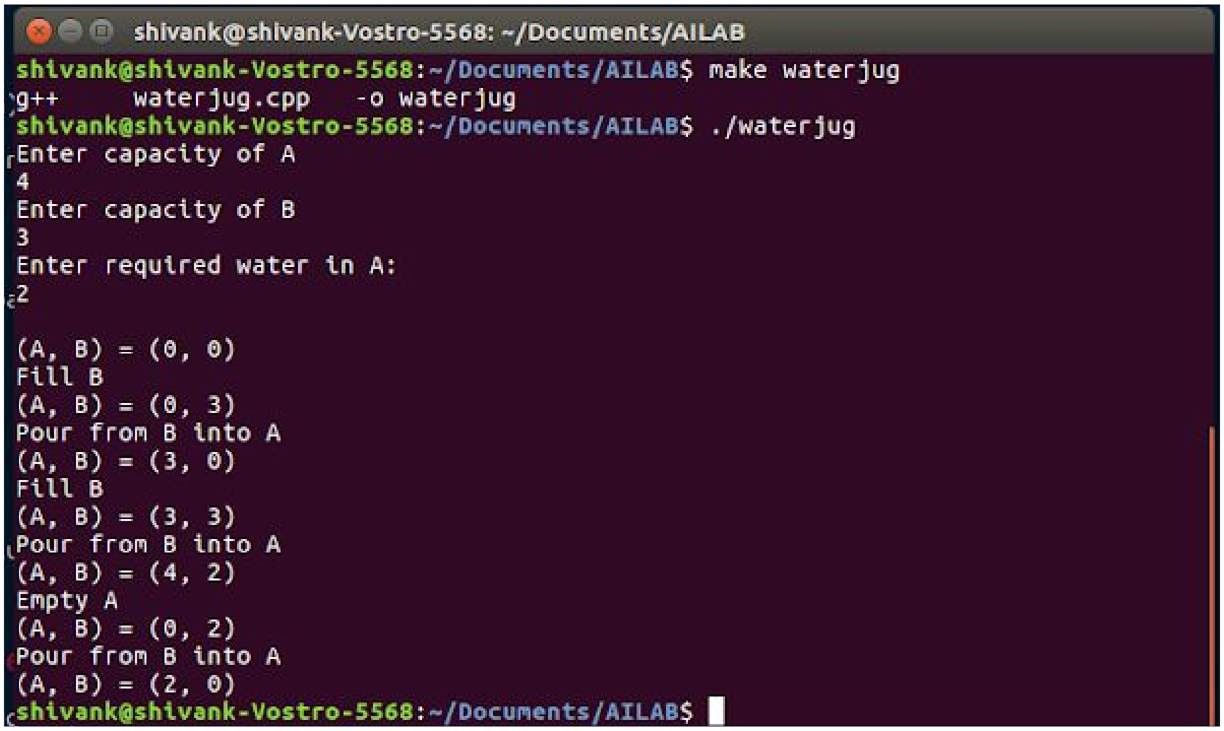
AIM : Program to solve Water Jug problem

CODE

```
#include <iostream.h>
using namespace std;
class Jug{

  int capacity;
  int value;
  public:Jug(int n)
  {
          capacity = n;
          value = 0;
  }
  void Fill()
  {
          value = capacity;
  }

void Empty()
{
          value = 0;
}
bool isFull()
{
          return value >= capacity;
}
bool isEmpty()
{
          return value == 0;
}
void operator[](Jug &B)
{
          int old_value = value;
          value = value + B.value;
          value = value > capacity?capacity:value;
          B.value = B.value - (value - old_value);
```

```
}
int getValue()
{
        return value;
}
};
int gcd(int n,int m)
{

        if(m<=n && n%m == 0)
        return m;
        if(n < m)
        return gcd(m,n);
        else
        return gcd(m,n%m);
}
bool check(int a,int b,int c){
        if(c>a){
                cout<<"A can't hold more water than it's capacity!\n";
                return false;
        }
        if(c % gcd(a,b) == 0)
        {
                return true;
        }
        cout<<"Can't reach this state with the given jugs\n";
        return false;
}
void solve(Jug A, Jug B, int result)
{
        while(A.getValue() != result)
        {
                if(!A.isFull() && B.isEmpty()){
                        cout<<"Fill B\n";
                        B.Fill();
                        cout << "(A, B) = (" << A.getValue() << ", " << B.getValue() << ")\n";
                }
                if(A.isFull()){
                        cout<<"Empty A\n";
                        A.Empty();
                        cout << "(A, B) = (" << A.getValue() << ", " << B.getValue() << ")\n";
                }
                cout<<"Pour from B into A\n";
                A[B];
                cout << "(A, B) = (" << A.getValue() << ", " << B.getValue() << ")\n";
    }

}
int main()
{
        int a, b, result;
        cout<<"Enter capacity of A\n";
        cin >> a;
        cout<<"Enter capacity of B\n";
        cin >> b;
        do{
                cout<<"Enter required water in A:\n";
                cin >> result;
        }
        while(!check(a,b,result));
        Jug A(a), B(b);
        cout << "\n(A, B) = (" << A.getValue() << ", " << B.getValue() << ")\n";
        solve(A, B, result);
        return 0;
}
```

OUTPUT



```
shivank@shivank-Vostro-5568: ~/Documents/AILAB
shivank@shivank-Vostro-5568:~/Documents/AILAB$ make waterjug
g++     waterjug.cpp   -o waterjug
shivank@shivank-Vostro-5568:~/Documents/AILAB$ ./waterjug
Enter capacity of A
4
Enter capacity of B
3
Enter required water in A:
2

(A, B) = (0, 0)
Fill B
(A, B) = (0, 3)
Pour from B into A
(A, B) = (3, 0)
Fill B
(A, B) = (3, 3)
Pour from B into A
(A, B) = (4, 2)
Empty A
(A, B) = (0, 2)
Pour from B into A
(A, B) = (2, 0)
shivank@shivank-Vostro-5568:~/Documents/AILAB$
```

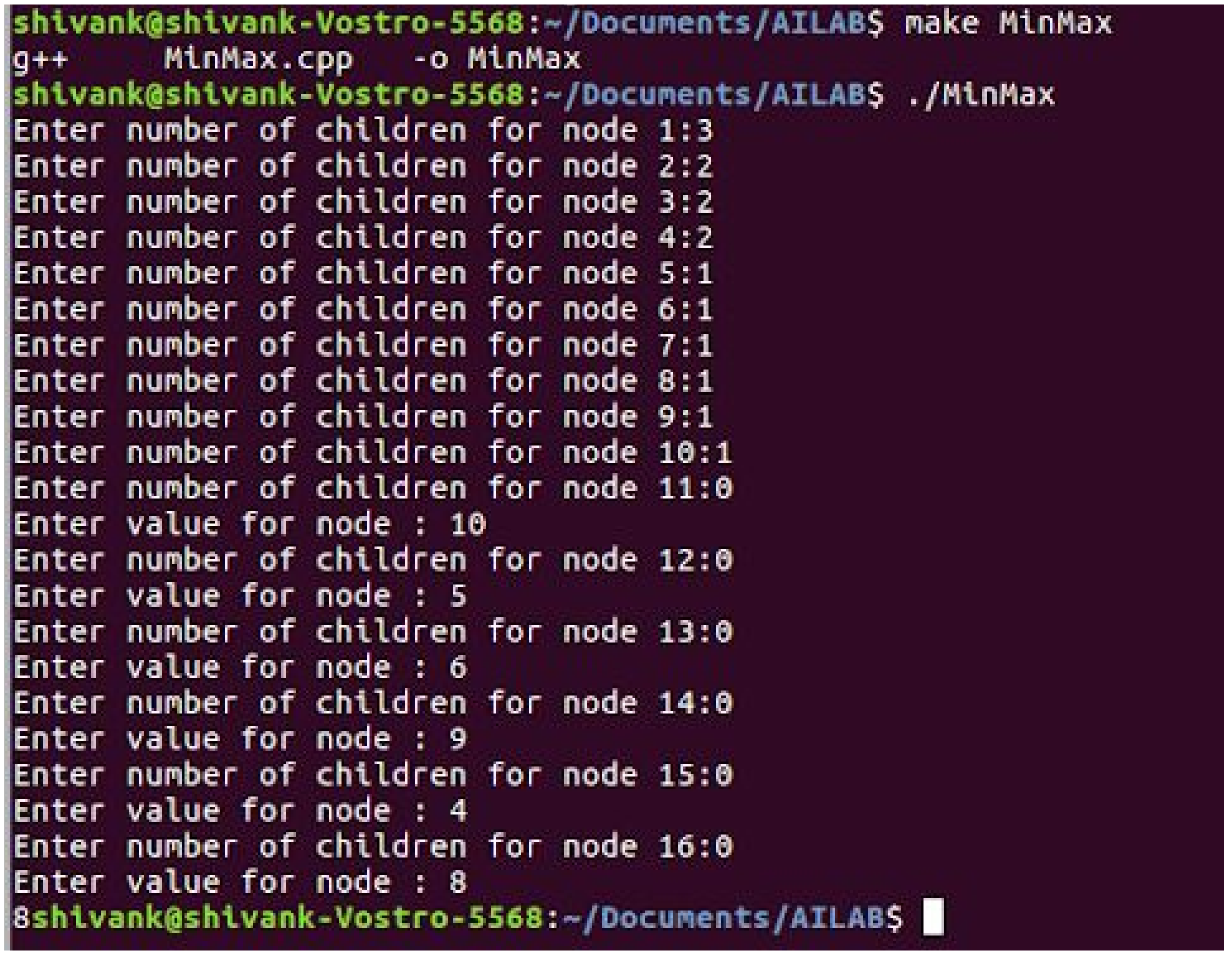
Practical 7

AIM : To Implement Min-Max algorithm

CODE

```
#include<bits/stdc++.h>
using namespace std;

struct Node
{
   int val;
   vector<Node *>child;
};

int minimax(Node *a,int minmax)
{
        if(a->child.empty())return a->val;
        int val;
        a->val=minimax(a->child[0],!minmax);
        for(int i=1;i<a->child.size();i++){
                val=minimax(a->child[i],!minmax);
                if(minmax==1){
                        if(val>a->val)a->val=val;
                }
                else{
                        if(val<a->val)a->val=val;
                }
        }
        return a->val;
}
void enter(Node *a){
        static int k=1;
        int n;
        cout<<"Enter number of children for node "<<k<<":";
        cin>>n;
        Node *newn;
        for(int i=0;i<n;i++){
                newn=new Node;
                a->child.push_back(newn);
			k++;
			enter(a->child[i]);
		}
		if(n==0){
			cout<<"Enter value for node : ";
			cin>>a->val;
		}
}
int main()
{
		Node *root;
		root = new Node;
		char ch='y';
		enter(root);
		cout<<minimax(root,1);
}
```

OUTPUT



```
shivank@shivank-Vostro-5568:~/Documents/AILAB$ make MinMax
g++     MinMax.cpp   -o MinMax
shivank@shivank-Vostro-5568:~/Documents/AILAB$ ./MinMax
Enter number of children for node 1:3
Enter number of children for node 2:2
Enter number of children for node 3:2
Enter number of children for node 4:2
Enter number of children for node 5:1
Enter number of children for node 6:1
Enter number of children for node 7:1
Enter number of children for node 8:1
Enter number of children for node 9:1
Enter number of children for node 10:1
Enter number of children for node 11:0
Enter value for node : 10
Enter number of children for node 12:0
Enter value for node : 5
Enter number of children for node 13:0
Enter value for node : 6
Enter number of children for node 14:0
Enter value for node : 9
Enter number of children for node 15:0
Enter value for node : 4
Enter number of children for node 16:0
Enter value for node : 8
8shivank@shivank-Vostro-5568:~/Documents/AILAB$
```

Practical 8

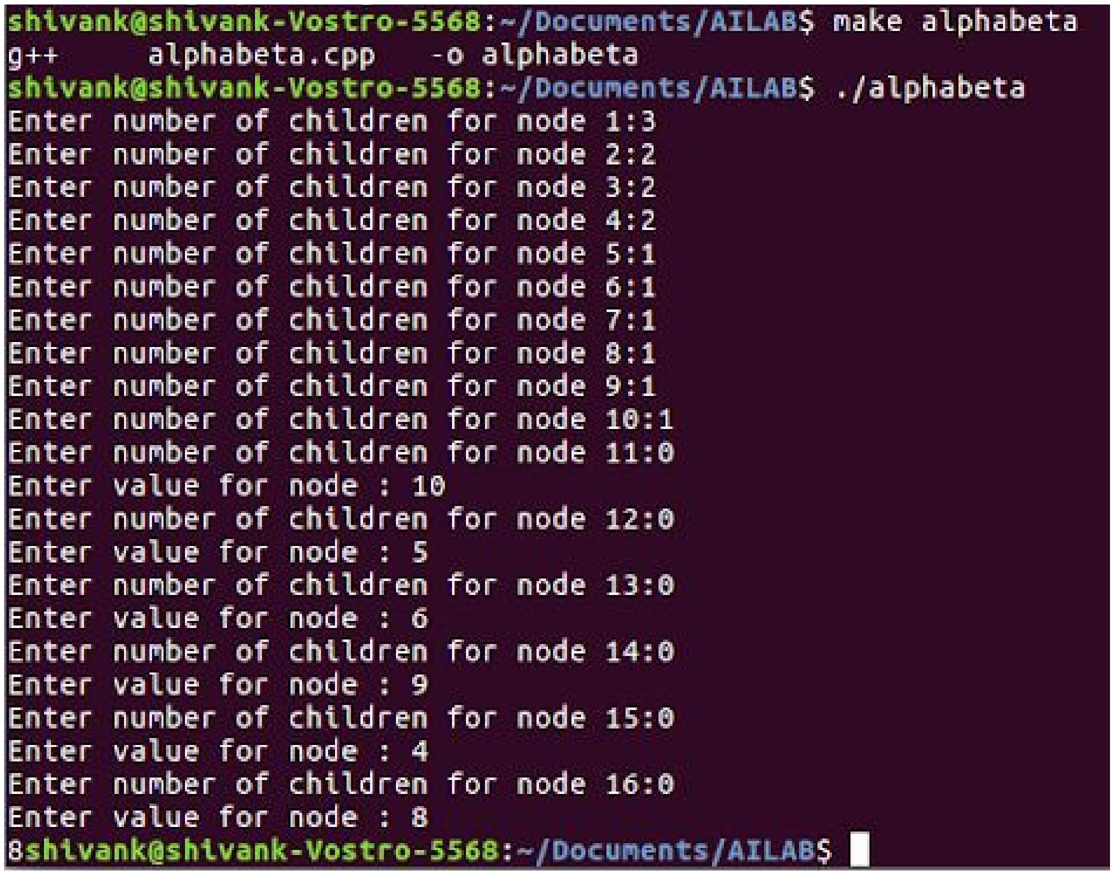
AIM : To Implement Min-Max algorithm with Alpha-Beta Pruning

CODE

```
#include<bits/stdc++.h>
#define INF 1000000
using namespace std;

struct Node
{
   int val;
   vector<Node *>child;
};

int max(int a, int b){
         return a>b?a:b;
}

int min(int a, int b){
         return a<b?a:b;
}

int minimax(Node *nn,int minmax,int a, int b)
{
         if(nn->child.empty())return nn->val;
         int val;
         nn->val=minimax(nn->child[0],!minmax,a,b);
         for(int i=1;i<nn->child.size();i++){
                  val=minimax(nn->child[i],!minmax,a,b);
                  if(minmax==1){
                           a=max(a,val);
                           if(a>=b)return b;
                           nn->val=max(nn->val,val);
                  }
                  else{
                           b=min(b,val);
                           if(a>=b)return a;
                           nn->val=min(nn->val,val);
```

```
                }
        }
        return nn->val;
}
void enter(Node *a){
        static int k=1;
        int n;
        cout<<"Enter number of children for node "<<k<<":";
        cin>>n;
        Node *newn;
        for(int i=0;i<n;i++){
                newn=new Node;
                a->child.push_back(newn);
                k++;
                enter(a->child[i]);
        }
        if(n==0){
                cout<<"Enter value for node : ";
                cin>>a->val;
        }
}

int main()
{
        int a,b;
        a=-1;
        b=INF;
        Node *root;
        root = new Node;
        char ch='y';
        enter(root);
        cout<<minimax(root,1,a,b);
}
```

OUTPUT



```
shivank@shivank-Vostro-5568:~/Documents/AILAB$ make alphabeta
g++     alphabeta.cpp   -o alphabeta
shivank@shivank-Vostro-5568:~/Documents/AILAB$ ./alphabeta
Enter number of children for node 1:3
Enter number of children for node 2:2
Enter number of children for node 3:2
Enter number of children for node 4:2
Enter number of children for node 5:1
Enter number of children for node 6:1
Enter number of children for node 7:1
Enter number of children for node 8:1
Enter number of children for node 9:1
Enter number of children for node 10:1
Enter number of children for node 11:0
Enter value for node : 10
Enter number of children for node 12:0
Enter value for node : 5
Enter number of children for node 13:0
Enter value for node : 6
Enter number of children for node 14:0
Enter value for node : 9
Enter number of children for node 15:0
Enter value for node : 4
Enter number of children for node 16:0
Enter value for node : 8
8shivank@shivank-Vostro-5568:~/Documents/AILAB$
```

Practical 9

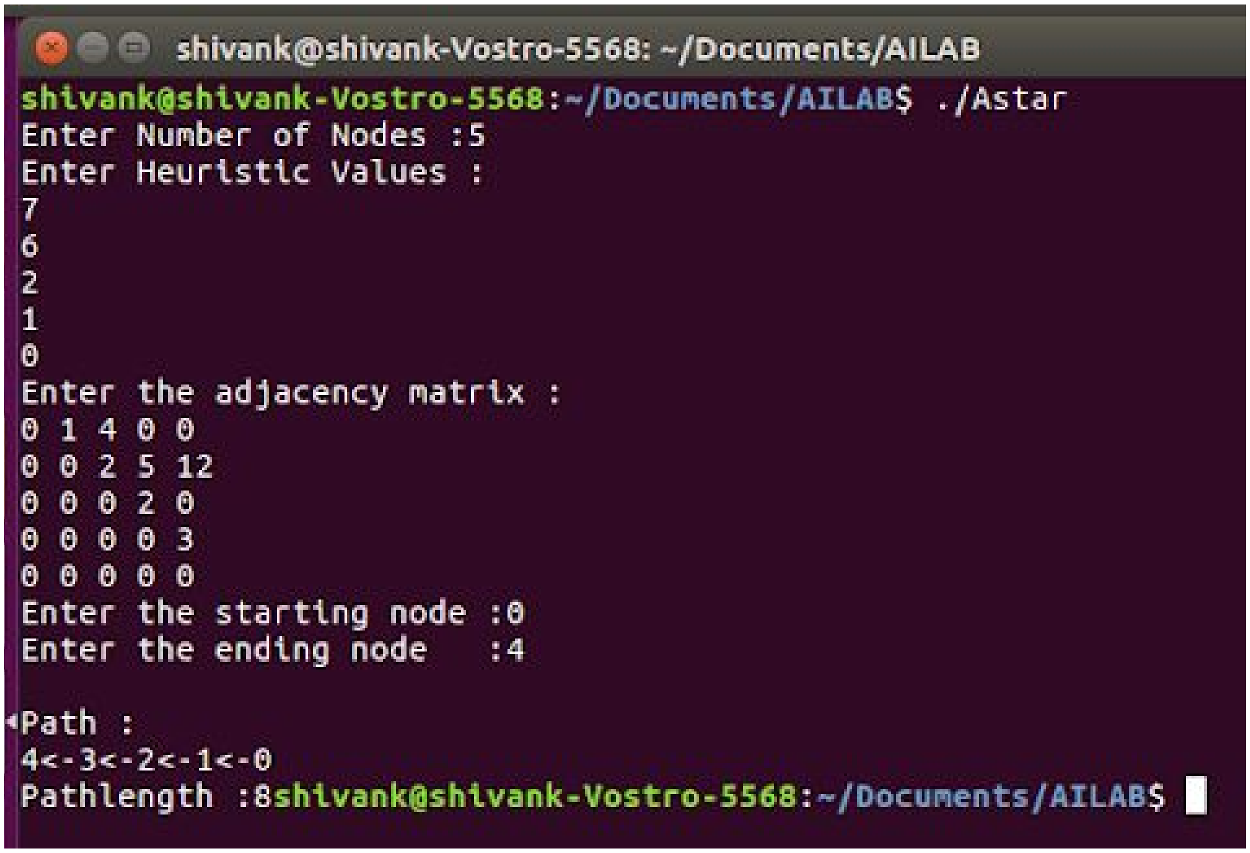
AIM : To Implement A* Algorithm to find the specified node

CODE

```
#include<bits/stdc++.h>
#define INF 1000000
using namespace std;

int main(){
	int n;
	cout<<"Enter Number of Nodes :";cin>>n;

	int h[n];
	cout<<"Enter Heuristic Values :\n";
	for(int i=0;i<n;i++)
	{
		cin>>h[i];
	}
	cout<<"Enter the adjacency matrix :\n";
	int a[n][n];
	for(int i=0;i<n;i++)
		for(int j=0;j<n;j++){
			cin>>a[i][j];
			if(a[i][j]==0)
				a[i][j]=INF;
		}

	int f[n][2],s,e,cur,min,t;
	bool done[n];
	cout<<"Enter the starting node :";cin>>s;
	cout<<"Enter the ending node   :";cin>>e;
	cur=s;
	for(int i=0;i<n;i++)
	{
		f[i][0]=INF;
		f[i][1]=-1;
		done[i]=false;
	}
```

```
        while(true)
        {
                //f(n) update
                done[cur]=true;
                for(int j=0;j<n;j++){
                        if(f[j][0]>a[cur][j]+h[j])
                        {
                                f[j][1]=cur;
                                f[j][0]=a[cur][j]+h[j];
                        }
                }

                min=f[0][0];t=-1;
                for(int j=1;j<n;j++){
                        if(min>=f[j][0]&& done[j]==false)
                        {
                                min=f[j][0];
                                t=j;
                        }
                }
                if(t==-1)break;
                cur=t;
        }
        cout<<endl;

        cout<<"Path :\n";
        int pathlength=0;
        while(f[e][1]!=-1)
        {
                cout<<e<<"<-";
                pathlength+=a[f[e][1]][e];
                e=f[e][1];
        }
        cout<<e;
        cout<<endl<<"Pathlength :"<<pathlength;

}
```

OUTPUT



```
shivank@shivank-Vostro-5568: ~/Documents/AILAB
shivank@shivank-Vostro-5568:~/Documents/AILAB$ ./Astar
Enter Number of Nodes :5
Enter Heuristic Values :
7
6
2
1
0
Enter the adjacency matrix :
0 1 4 0 0
0 0 2 5 12
0 0 0 2 0
0 0 0 0 3
0 0 0 0 0
Enter the starting node :0
Enter the ending node   :4

Path :
4<-3<-2<-1<-0
Pathlength :8shivank@shivank-Vostro-5568:~/Documents/AILAB$ 
```

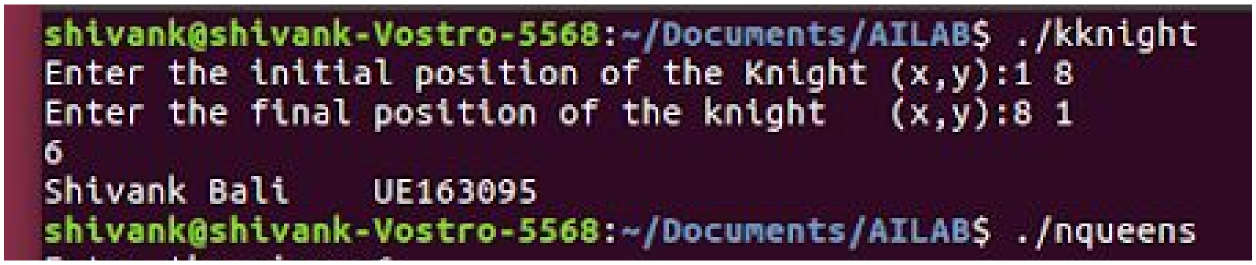
Practical 10

AIM : To solve the Knight Problem

CODE

```
#include<bits/stdc++.h>
using namespace std;

struct cell
{
	int x, y;
	int dis;
	cell() {}
	cell(int x, int y, int dis) : x(x), y(y), dis(dis) {}
};

bool isInside(int x, int y, int N)
{
	if (x >= 1 && x <= N && y >= 1 && y <= N)
		return true;
	return false;
}

int minStepToReachTarget(int knightPos[], int targetPos[], int N)
{
	int dx[] = {-2, -1, 1, 2, -2, -1, 1, 2};
	int dy[] = {-1, -2, -2, -1, 1, 2, 2, 1};
	queue<cell> q;

	q.push(cell(knightPos[0], knightPos[1], 0));
	cell t;
	int x, y;
	bool visit[N + 1][N + 1];

	for (int i = 1; i <= N; i++)
		for (int j = 1; j <= N; j++)
			visit[i][j] = false;

	visit[knightPos[0]][knightPos[1]] = true;
	while (!q.empty())
	{
		t = q.front();
		q.pop();

		if (t.x == targetPos[0] && t.y == targetPos[1])
			return t.dis;

		for (int i = 0; i < 8; i++)
		{
			x = t.x + dx[i];
			y = t.y + dy[i];

			if (isInside(x, y, N) && !visit[x][y]) {
				visit[x][y] = true;
				q.push(cell(x, y, t.dis + 1));
			}
		}
	}
}

int main()
{
	int N = 8;
	int x1,y1,x2,y2;
	cout<<"Enter the initial position of the Knight (x,y):";
	cin>>x1>>y1;
	cout<<"Enter the final position of the knight   (x,y):";
	cin>>x2>>y2;
	int knightPos[] = {x1, y1};
	int targetPos[] = {x2, y2};
	cout << minStepToReachTarget(knightPos, targetPos, N)<<endl;
	cout<<"Shivank Bali\tUE163095\n" ;
	return 0;
}
```

OUTPUT



```
shivank@shivank-Vostro-5568:~/Documents/AILAB$ ./kknight
Enter the initial position of the Knight (x,y):1 8
Enter the final position of the knight   (x,y):8 1
6
Shivank Bali    UE163095
shivank@shivank-Vostro-5568:~/Documents/AILAB$ ./nqueens
```

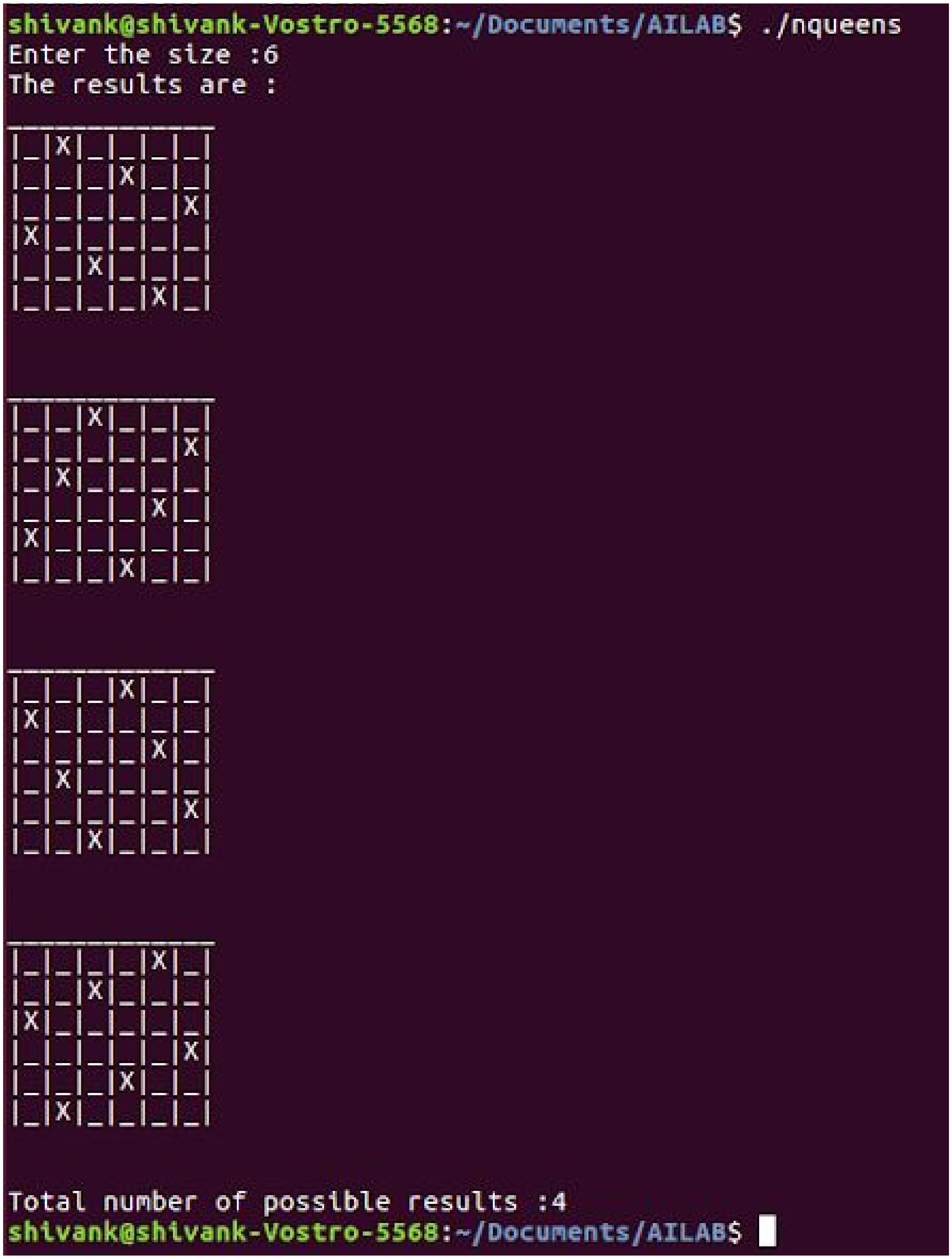
Practical 11

AIM : To solve the N-Queens Problem

CODE

```
#include<iostream>
#include<cmath>
using namespace std;

bool check(int a, int b, int c ,int d){
	if(b==d||abs(a-c)==abs(b-d))
		return false;
	return true;
}

void qprint(int x[],int n)
{
	for(int j=0;j<n;j++)
		cout<<"__";
	cout<<"_"<<endl;
	for(int i=0;i<n;i++)
	{
		cout<<'|';
		for(int j=0;j<n;j++)
		{
			if(x[i]==j)
				cout<<'X'<<'|';
			else
				cout<<"_|";
		}
		cout<<endl;

	}
	cout<<endl<<endl;
}

int nqueen(int x[],int n,int k)
{
	int count=0;
	for(int i=0;i<k-1;i++)
	for(int j=i+1;j<k;j++)
		if(check(i,x[i],j,x[j])==false)
			return 0;

	if(k==n)
		{
			qprint(x,n);
			count++;
		}
	else{
		for(int i=0;i<n;i++)
		{
			x[k]=i;
			count+=nqueen(x,n,k+1);
		}
	}
	return count;
}

int main()
{
	int n;
	cout<<"Enter the size :";
	cin>>n;
	int a[n];
	cout<<"The results are :\n";
	cout<<"Total number of possible results :"<<nqueen(a,n,0)<<endl;
	return 0;
}
```

OUTPUT



```
shivank@shivank-Vostro-5568:~/Documents/AILAB$ ./nqueens
Enter the size :6
The results are :

_____________
|_|X|_|_|_|_|
|_|_|_|X|_|_|
|_|_|_|_|_|X|
|X|_|_|_|_|_|
|_|_|X|_|_|_|
|_|_|_|_|X|_|


_____________
|_|_|X|_|_|_|
|_|_|_|_|_|X|
|_|X|_|_|_|_|
|_|_|_|_|X|_|
|X|_|_|_|_|_|
|_|_|_|X|_|_|


_____________
|_|_|_|X|_|_|
|X|_|_|_|_|_|
|_|_|_|_|X|_|
|_|X|_|_|_|_|
|_|_|_|_|_|X|
|_|_|X|_|_|_|


_____________
|_|_|_|_|X|_|
|_|_|X|_|_|_|
|X|_|_|_|_|_|
|_|_|_|_|_|X|
|_|_|_|X|_|_|
|_|X|_|_|_|_|


Total number of possible results :4
shivank@shivank-Vostro-5568:~/Documents/AILAB$
```