

INDEX

S. No.	Experiment	Date	Sign & Remark
01.	Write a program to solve the 8-Puzzle problem using Generate and Test Strategy.		
02.	Write a program to solve the 8-Puzzle problem using DFID Technique.		
03.	Write a program to solve the 3-SAT Problem using Variable Neighbourhood Descent Algorithm.		
04.	Write a program to solve the 3-SAT Problem using Stochastic Hill Climbing Algorithm.		
05.	Write a program to solve the 8-Puzzle problem using A* algorithm.		
06.	WAP to find maximum of two/three numbers.		
07.	WAP to find factorial of a number.		
08.	WAP to find sum of first N numbers.		

INDEX

S. No.	Experiment	Date	Sign & Remark
09.	Write a program to find Fibonacci sequence upto Nth term.		
10.	Write a program to represent following statements in Prolog. 1) Prakash likes food if the food is edible and it tastes sweet. 2) Chocolates tastes sweet. 3) Toffees tastes sweet. 4) Gourd tastes bitter. 5) Chocolates are edible. 6) Toffees are edible. 7) Gourd is edible.		
11.	Write a program to solve AND OR Graph using AO* Search Algorithm.		

Experiment 1

Aim:

Write a program to solve the 8-Puzzle problem using Generate and Test Strategy.

You can take the current state as:

8	1	3
4		5
2	7	6

And the final state must be:

1	2	3
8		4
7	6	5

Code:

```
//Aneesh Panchal
//2K20/MC/21

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

struct Node{
    vector<vector<int>> data;
    Node *n1, *n2, *n3, *n4;
};

Node *createNode(vector<vector<int>> data){
    Node *node = new Node;
    node->data = data;
    node->n1 = node->n2 = node->n3 = node->n4 = NULL;
    return node;
}

vector<vector<int>> moveUp(vector<vector<int>> data){
    int i, j;
    for(i = 0; i < 3; i++){
        for(j = 0; j < 3; j++){
            if(data[i][j] == 0){
                if(i == 0)
                    return data;
                else{
                    data[i][j] = data[i-1][j];
                    data[i-1][j] = 0;
                    return data;
                }
            }
        }
    }
}
```

```

}

vector<vector<int>>> moveDown(vector<vector<int>>> data){
    int i, j;
    for(i = 0; i < 3; i++){
        for(j = 0; j < 3; j++){
            if(data[i][j] == 0){
                if(i == 2)
                    return data;
                else{
                    data[i][j] = data[i+1][j];
                    data[i+1][j] = 0;
                    return data;
                }
            }
        }
    }
}

```

```

vector<vector<int>>> moveLeft(vector<vector<int>>> data){
    int i, j;
    for(i = 0; i < 3; i++){
        for(j = 0; j < 3; j++){
            if(data[i][j] == 0){
                if(j == 0)
                    return data;
                else{
                    data[i][j] = data[i][j-1];
                    data[i][j-1] = 0;
                    return data;
                }
            }
        }
    }
}

```

```

vector<vector<int>>> moveRight(vector<vector<int>>> data){
    int i, j;
    for(i = 0; i < 3; i++){
        for(j = 0; j < 3; j++){
            if(data[i][j] == 0){
                if(j == 2)
                    return data;
                else{
                    data[i][j] = data[i][j+1];
                    data[i][j+1] = 0;
                    return data;
                }
            }
        }
    }
}

```

```

vector<vector<int>> findSol(vector<vector<int>> start, vector<vector<int>> goal){
    queue<Node*> q;
    Node *root = createNode(start);
    q.push(root);
    int i=0;
    while(!q.empty()){
        Node *temp = q.front();
        q.pop();
        ++i;
        if(temp->data == goal){
            cout<<"Solution found after checking "<<i<<" states"<<endl;
            cout<<"Goal State Reached"<<endl;
            return temp->data;
        }
        else{
            vector<vector<int>> up = moveUp(temp->data);
            vector<vector<int>> down = moveDown(temp->data);
            vector<vector<int>> left = moveLeft(temp->data);
            vector<vector<int>> right = moveRight(temp->data);
            if(up != temp->data){
                Node *node = createNode(up);
                temp->n1 = node;
                q.push(node);
            }
            if(down != temp->data){
                Node *node = createNode(down);
                temp->n2 = node;
                q.push(node);
            }
            if(left != temp->data){
                Node *node = createNode(left);
                temp->n3 = node;
                q.push(node);
            }
            if(right != temp->data){
                Node *node = createNode(right);
                temp->n4 = node;
                q.push(node);
            }
        }
    }
}

int main(){
    vector<vector<int>> start = {{8,1,3},{4,0,5},{2,7,6}};
    vector<vector<int>> goal = {{1,2,3},{8,0,4},{7,6,5}};
    vector<vector<int>> sol = findSol(start, goal);
    for(int i = 0; i < 3; i++){
        for(int j = 0; j < 3; j++){
            cout<<sol[i][j]<<" ";
        }
        cout<<endl;
    }
}

```

```
    return 0;  
}
```

Output:

```
PROBLEMS  OUTPUT  TERMINAL  SQL CONSOLE  DEBUG CONSOLE  
  
PS E:\Codes\MC312> cd "e:\Codes\MC312\" ; if ($?) { g++ Exp1.cpp -o Exp1 } ; if ($?) { .\Exp1 }  
Solution found after checking 56229 states  
Goal State Reached  
1 2 3  
8 0 4  
7 6 5  
  
PS E:\Codes\MC312> █
```

Experiment 2

Aim:

Write a program to solve the 8-Puzzle problem using DFID Technique.

You can take the current state as:

8	1	3
4		5
2	7	6

And the final state must be:

1	2	3
8		4
7	6	5

Code:

```
//Aneesh Panchal
//2K20/MC/21

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

struct Node{
    vector<vector<int>> data;
    Node *n1, *n2, *n3, *n4;
};

Node *createNode(vector<vector<int>> data){
    Node *node = new Node;
    node->data = data;
    node->n1 = node->n2 = node->n3 = node->n4 = NULL;
    return node;
}

vector<vector<int>> moveUp(vector<vector<int>> data){
    int i, j;
    for(i = 0; i < 3; i++){
        for(j = 0; j < 3; j++){
            if(data[i][j] == 0){
                if(i == 0)
                    return data;
                else{
                    data[i][j] = data[i-1][j];
                    data[i-1][j] = 0;
                    return data;
                }
            }
        }
    }
}
```

```

}

vector<vector<int>>> moveDown(vector<vector<int>>> data){
    int i, j;
    for(i = 0; i < 3; i++){
        for(j = 0; j < 3; j++){
            if(data[i][j] == 0){
                if(i == 2)
                    return data;
                else{
                    data[i][j] = data[i+1][j];
                    data[i+1][j] = 0;
                    return data;
                }
            }
        }
    }
}

```

```

vector<vector<int>>> moveLeft(vector<vector<int>>> data){
    int i, j;
    for(i = 0; i < 3; i++){
        for(j = 0; j < 3; j++){
            if(data[i][j] == 0){
                if(j == 0)
                    return data;
                else{
                    data[i][j] = data[i][j-1];
                    data[i][j-1] = 0;
                    return data;
                }
            }
        }
    }
}

```

```

vector<vector<int>>> moveRight(vector<vector<int>>> data){
    int i, j;
    for(i = 0; i < 3; i++){
        for(j = 0; j < 3; j++){
            if(data[i][j] == 0){
                if(j == 2)
                    return data;
                else{
                    data[i][j] = data[i][j+1];
                    data[i][j+1] = 0;
                    return data;
                }
            }
        }
    }
}

```



```

bool DFS(Node *root, vector<vector<int>> data){
    if(root == NULL)
        return false;
    if(root->data == data)
        return true;
    return DFS(root->n1, data) || DFS(root->n2, data) || DFS(root->n3, data) || DFS(root->n4, data);
}

vector<vector<int>> findSol(vector<vector<int>> start, vector<vector<int>> goal){
    queue<Node*> q1, q2;
    Node *root = createNode(start);
    q2.push(root);
    int i=0;
    while(true){
        int j = q2.size();
        for(int k=0; k<j; k++){
            Node *temp = q2.front();
            q2.pop();
            q1.push(temp);
        }

        while(!q1.empty()){
            Node *temp = q1.front();
            q1.pop();
            vector<vector<int>> up = moveUp(temp->data);
            vector<vector<int>> down = moveDown(temp->data);
            vector<vector<int>> left = moveLeft(temp->data);
            vector<vector<int>> right = moveRight(temp->data);
            if(up != temp->data){
                Node *node = createNode(up);
                temp->n1 = node;
                q2.push(node);
            }
            if(down != temp->data){
                Node *node = createNode(down);
                temp->n2 = node;
                q2.push(node);
            }
            if(left != temp->data){
                Node *node = createNode(left);
                temp->n3 = node;
                q2.push(node);
            }
            if(right != temp->data){
                Node *node = createNode(right);
                temp->n4 = node;
                q2.push(node);
            }
        }

        if(DFS(root, goal)){
            cout<<"Solution found at depth "<<i<<endl;

```

```

        cout<<"Goal State Reached"<<endl;
        return goal;
    }
    else
        i++;
}
}

int main(){
    vector<vector<int>> start = {{8,1,3},{4,0,5},{2,7,6}};
    vector<vector<int>> goal = {{1,2,3},{8,0,4},{7,6,5}};
    vector<vector<int>> sol = findSol(start, goal);
    for(int i = 0; i < 3; i++){
        for(int j = 0; j < 3; j++){
            cout<<sol[i][j]<<" ";
        }
        cout<<endl;
    }
    return 0;
}

```

Output:

```

PROBLEMS  OUTPUT  TERMINAL  SQL CONSOLE  DEBUG CONSOLE
PS E:\Codes\MC312> cd "e:\Codes\MC312\" ; if ($?) { g++ Exp2.cpp -o Exp2 } ; if ($?) { .\Exp2 }
Solution found at depth 9
Goal State Reached
1 2 3
8 0 4
7 6 5

PS E:\Codes\MC312>

```

Experiment 3

Aim:

Write a program to solve the 3- SAT Problem using Variable Neighbourhood Descent Algorithm.

$F = (b \vee c') \wedge (c \vee d') \wedge (b') \wedge (a' \vee e') \wedge (c' \vee e) \wedge (c' \vee d')$.

The initial state is $\{a=1, b=1, c=1, d=1, e=1\}$

Code:

```
//Aneesh Panchal
//2K20/MC/21

#include<iostream>
#include<vector>
using namespace std;
vector<int> num;

int heuristicValue(vector<int> data){
    int a = data[0], b = data[1], c = data[2], d = data[3], e = data[4];
    int i = b || (!c);
    int ii = c || (!d);
    int iii = !b;
    int iv = (!a) || (!e);
    int v = e || (!c);
    int vi = (!c) || (!d);
    int val = i + ii + iii + iv + v + vi;
    return val;
}

vector<int> NeighbourFunction(vector<int> data, int k){
    vector<int> temp = data;
    vector<int> sol = data;
    int a = k%10;
    int b = (k/10)%10;
    int c = (k/100)%10;
    int d = (k/1000)%10;
    int e = (k/10000)%10;
    int h = heuristicValue(data);
    int ht = h;

    if(a==1 || b==1 || c==1 || d==1 || e==1){
        for(int i=0; i<5; i++){
            temp = data;
            temp[i] = !temp[i];
            ht = heuristicValue(temp);
            if(ht > h){
                sol = temp;
                h = ht;
            }
        }
    }
}
```

```

if(a==2 || b==2 || c==2 || d==2 || e==2){
    for(int i=0; i<5; i++){
        for(int j=i+1; j<5; j++){
            temp = data;
            temp[i] = !temp[i];
            temp[j] = !temp[j];
            ht = heuristicValue(temp);
            if(ht > h){
                sol = temp;
                h = ht;
            }
        }
    }
}

if(a==3 || b==3 || c==3 || d==3 || e==3){
    for(int i=0; i<5; i++){
        for(int j=i+1; j<5; j++){
            for(int k=j+1; k<5; k++){
                temp = data;
                temp[i] = !temp[i];
                temp[j] = !temp[j];
                temp[k] = !temp[k];
                ht = heuristicValue(temp);
                if(ht > h){
                    sol = temp;
                    h = ht;
                }
            }
        }
    }
}

if(a==4 || b==4 || c==4 || d==4 || e==4){
    for(int i=0; i<5; ++i){
        temp = data;
        for(int j=0; j<5; ++j){
            if(i==j)
                continue;
            else
                temp[j] = !temp[j];
        }
        ht = heuristicValue(temp);
        if(ht > h){
            sol = temp;
            h = ht;
        }
    }
}

if(a==5 || b==5 || c==5 || d==5 || e==5){
    temp = data;

```

```

        temp[0] = !temp[0];
        temp[1] = !temp[1];
        temp[2] = !temp[2];
        temp[3] = !temp[3];
        temp[4] = !temp[4];
        ht = heuristicValue(temp);
        if(ht > h){
            sol = temp;
            h = ht;
        }
    }
    return sol;
}

void Combi(vector<int> vect, int reqLen, int start, int currLen, bool check[], int last){
    int x = 0;
    if(currLen > reqLen)
        return;
    else if(currLen == reqLen){
        for(int i=0; i<last; i++){
            if(check[i] == true){
                x = vect[i] + 10*x;
            }
        }
        num.push_back(x);
        return;
    }
    if(start==last)
        return;
    check[start] = true;
    Combi(vect, reqLen, start+1, currLen+1, check, last);
    check[start] = false;
    Combi(vect, reqLen, start+1, currLen, check, last);
}

vector<int> findSol(vector<int> data){
    vector<int> index = {1,2,3,4,5};
    int n = data.size();
    int k = 1;
    bool check[5];
    for(int i=0; i<data.size(); i++)
        check[i] = false;
    while(true){
        Combi(index, k, 0, 0, check, n);
        for(int i=0; i<num.size(); i++){
            data = NeighbourFunction(data, num[i]);
            if(heuristicValue(data) == 6)
                return data;
        }
        num.clear();
        ++k;
    }
    return data;
}

```

```

}

int main(){
    vector<int> init_data = {1,1,1,1,1};
    vector<int> soln = findSol(init_data);
    cout<<endl<<"Initial Data: "<<endl;
    cout<<"a b c d e"<<endl;
    for(int i=0; i<init_data.size(); i++)
        cout<<init_data[i]<<" ";
    cout<<endl<<"Heuristic Value: "<<heuristicValue(init_data)<<endl;
    cout<<endl<<"Solution: "<<endl;
    cout<<"a b c d e"<<endl;
    for(int i=0; i<soln.size(); i++)
        cout<<soln[i]<<" ";
    cout<<endl<<"Heuristic Value: "<<heuristicValue(soln)<<endl<<endl;
    return 0;
}

```

Output:

```

PROBLEMS  OUTPUT  TERMINAL  SQL CONSOLE  DEBUG CONSOLE

PS E:\Codes\MC312> cd "e:\Codes\MC312\" ; if ($?) { g++ Exp3.cpp -o Exp3 } ; if ($?) { .\Exp3 }

Initial Data:
a b c d e
1 1 1 1 1
Heuristic Value: 3

Solution:
a b c d e
1 0 0 0 0
Heuristic Value: 6

PS E:\Codes\MC312>

```

Experiment 4

Aim:

Write a program to solve the 3- SAT Problem using Stochastic Hill Climbing Algorithm.

$F = (b \vee c') \wedge (c \vee d') \wedge (b') \wedge (a' \vee e') \wedge (c' \vee e) \wedge (c' \vee d')$.

The initial state is $\{a=1, b=1, c=1, d=1, e=1\}$

Code:

```
//Aneesh Panchal
//2K20/MC/21

#include<iostream>
#include<vector>
#include<queue>
#include<cmath>
using namespace std;

struct Node{
    vector<int> data;
    Node *n1, *n2;
    int same, changed;
    int depth;
};

Node *createNode(vector<int> data, int depth){
    Node *node = new Node;
    node->data = data;
    node->same = node->changed = 0;
    node->n1 = node->n2 = NULL;
    node->depth = depth;
    return node;
}

int heuristicValue(vector<int> data){
    int a = data[0], b = data[1], c = data[2], d = data[3], e = data[4];
    int i = b || (!c);
    int ii = c || (!d);
    int iii = !b;
    int iv = (!a) || (!e);
    int v = e || (!c);
    int vi = (!c) || (!d);
    int val = i + ii + iii + iv + v + vi;
    return val;
}

float Probability(int h, int hp){
    int e = exp(-(hp-h)/10);
    float p = (float)1/(1 + e);
    return p;
}
```

```

vector<int> NeighbourFunction(vector<int> data, int index){
    data[index] = !data[index];
    return data;
}

vector<int> findSol(vector<int> arr){
    queue<Node*> q;
    Node *root = createNode(arr, 0);
    q.push(root);
    vector<int> same = arr;
    vector<int> changed = arr;
    int hp = heuristicValue(arr);
    int h = heuristicValue(arr);
    int deep = 0;
    float prob;
    while(true){
        Node *temp = q.front();
        q.pop();
        if(heuristicValue(temp->data) == 6){
            cout<<"Goal State Reached"<<endl;
            return temp->data;
        }
        else{
            deep = temp->depth;
            same = temp->data;
            changed = NeighbourFunction(same, deep);
            hp = heuristicValue(same);
            h = heuristicValue(changed);
            prob = Probability(h, hp);
            if(prob >= 0.5){
                temp->n2 = createNode(changed, deep+1);
                temp->n2->changed = 1;
                q.push(temp->n2);
            }
            if(prob <= 0.5){
                temp->n1 = createNode(same, deep+1);
                temp->n1->same = 1;
                q.push(temp->n1);
            }
        }
    }
}

int main(){
    vector<int> init_data = {1,1,1,1,1};
    vector<int> soln = findSol(init_data);
    cout<<endl<<"Initial Data: "<<endl;
    cout<<"a b c d e"<<endl;
    for(int i=0; i<init_data.size(); i++)
        cout<<init_data[i]<<" ";
    cout<<endl<<"Heuristic Value: "<<heuristicValue(init_data)<<endl;
    cout<<endl<<"Solution: "<<endl;
}

```



```

    cout<<"a b c d e"<<endl;
    for(int i=0; i<soln.size(); i++)
        cout<<soln[i]<<" ";
    cout<<endl<<"Heuristic Value: "<<heuristicValue(soln)<<endl<<endl;
    return 0;
}

```

Output:

```

PROBLEMS  OUTPUT  TERMINAL  SQL CONSOLE  DEBUG CONSOLE

PS E:\Codes\MC312> cd "e:\Codes\MC312\" ; if ($?) { g++ Exp4.cpp -o Exp4 } ; if ($?) { .\Exp4 }
Goal State Reached

Initial Data:
a b c d e
1 1 1 1 1
Heuristic Value: 3

Solution:
a b c d e
0 0 0 0 1
Heuristic Value: 6

PS E:\Codes\MC312>

```

Experiment 5

Aim:

Write a program to solve the 8-Puzzle problem using A* algorithm.

You can take the current state as:

2	8	3
1	6	4
7		5

And the final state must be:

1	2	3
8		4
7	6	5

Code:

```
//Aneesh Panchal
//2K20/MC/21

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

struct Node{
    vector<vector<int>> data;
    Node *n1, *n2, *n3, *n4;
    int depth;
};

Node *createNode(vector<vector<int>> data, int depth){
    Node *node = new Node;
    node->data = data;
    node->n1 = node->n2 = node->n3 = node->n4 = NULL;
    node->depth = depth;
    return node;
}

vector<vector<int>> moveUp(vector<vector<int>> data){
    int i, j;
    for(i = 0; i < 3; i++){
        for(j = 0; j < 3; j++){
            if(data[i][j] == 0){
                if(i == 0)
                    return data;
                else{
                    data[i][j] = data[i-1][j];
                    data[i-1][j] = 0;
                    return data;
                }
            }
        }
    }
}
```

```

    }
}

vector<vector<int>> moveDown(vector<vector<int>> data){
    int i, j;
    for(i = 0; i < 3; i++){
        for(j = 0; j < 3; j++){
            if(data[i][j] == 0){
                if(i == 2)
                    return data;
                else{
                    data[i][j] = data[i+1][j];
                    data[i+1][j] = 0;
                    return data;
                }
            }
        }
    }
}

```

```

vector<vector<int>> moveLeft(vector<vector<int>> data){
    int i, j;
    for(i = 0; i < 3; i++){
        for(j = 0; j < 3; j++){
            if(data[i][j] == 0){
                if(j == 0)
                    return data;
                else{
                    data[i][j] = data[i][j-1];
                    data[i][j-1] = 0;
                    return data;
                }
            }
        }
    }
}

```

```

vector<vector<int>> moveRight(vector<vector<int>> data){
    int i, j;
    for(i = 0; i < 3; i++){
        for(j = 0; j < 3; j++){
            if(data[i][j] == 0){
                if(j == 2)
                    return data;
                else{
                    data[i][j] = data[i][j+1];
                    data[i][j+1] = 0;
                    return data;
                }
            }
        }
    }
}

```

```

}

int h(vector<vector<int>> data, vector<vector<int>> goal = {{1,2,3},{8,0,4},{7,6,5}}){
    int count = 0;
    for(int i = 0; i < 3; i++){
        for(int j = 0; j < 3; j++){
            if(data[i][j] != goal[i][j])
                count++;
        }
    }
    return count;
}

int fval(int gval, int hval){
    return gval + hval;
}

struct cmp{
    inline bool operator() (Node* a, Node* b){
        return a->depth + h(a->data) > b->depth + h(b->data);
    }
};

void printmat(Node *node){
    for(int i = 0; i < 3; i++){
        for(int j = 0; j < 3; j++){
            cout<<node->data[i][j]<<" ";
        }
        cout<<endl;
    }
    cout<<"f() = h() + g(): "<<fval(node->depth,h(node->data))<<endl;
    cout<<endl;
}

vector<vector<int>> findSol(vector<vector<int>> start, vector<vector<int>> goal){
    priority_queue<Node*, vector<Node*>, cmp> q;
    Node *root = createNode(start,0);
    q.push(root);
    int i=0;
    int deep = 0;
    int min = 0;
    while(!q.empty()){
        Node *temp = q.top();
        q.pop();
        ++i;
        printmat(temp);
        min = fval(temp->depth, h(temp->data));
        if((temp->data == goal) && (min <= fval(q.top()->depth, h(q.top()->data)))){
            cout<<"Number of nodes expanded: "<<i<<endl;
            cout<<"Solution found after "<<temp->depth<<" moves"<<endl;
            cout<<"Goal State Reached"<<endl;
            return temp->data;
        }
    }
}

```

```

else{
    deep = temp->depth;
    vector<vector<int>> up = moveUp(temp->data);
    vector<vector<int>> down = moveDown(temp->data);
    vector<vector<int>> left = moveLeft(temp->data);
    vector<vector<int>> right = moveRight(temp->data);
    if(up != temp->data){
        Node *node = createNode(up,deep+1);
        temp->n1 = node;
        q.push(node);
    }
    if(down != temp->data){
        Node *node = createNode(down,deep+1);
        temp->n2 = node;
        q.push(node);
    }
    if(left != temp->data){
        Node *node = createNode(left,deep+1);
        temp->n3 = node;
        q.push(node);
    }
    if(right != temp->data){
        Node *node = createNode(right,deep+1);
        temp->n4 = node;
        q.push(node);
    }
}
}
}

int main(){
    vector<vector<int>> start = {{2,8,3},{1,6,4},{7,0,5}};
    vector<vector<int>> goal = {{1,2,3},{8,0,4},{7,6,5}};
    vector<vector<int>> sol = findSol(start, goal);
    for(int i = 0; i < 3; i++){
        for(int j = 0; j < 3; j++){
            cout<<sol[i][j]<<" ";
        }
        cout<<endl;
    }
    return 0;
}

```

Output:

```
PROBLEMS  OUTPUT  TERMINAL  SQL CONSOLE  DEBUG CONSOLE

PS E:\Codes\MC312> cd "e:\Codes\MC312\" ; if ($?) { g++ Exp5.cpp -o Exp5 } ; if ($?) { .\Exp5 }

2 8 3
1 6 4
7 0 5
f() = h() + g(): 5

2 8 3
1 0 4
7 6 5
f() = h() + g(): 4

2 0 3
1 8 4
7 6 5
f() = h() + g(): 6

2 8 3
0 1 4
7 6 5
f() = h() + g(): 6

2 8 3
1 0 4
7 6 5
f() = h() + g(): 6

0 2 3
1 8 4
7 6 5
f() = h() + g(): 6

2 8 3
1 0 4
7 6 5
f() = h() + g(): 6

1 2 3
0 8 4
7 6 5
f() = h() + g(): 6

1 2 3
8 0 4
7 6 5
f() = h() + g(): 5

Number of nodes expanded: 9
Solution found after 5 moves
Goal State Reached
1 2 3
8 0 4
7 6 5

PS E:\Codes\MC312>
```

Experiment 6

Aim:

WAP to find maximum of two/three numbers.

Code:

```
find_max2(X, Y, X) :- X >= Y, !.
```

```
find_max2(X, Y, Y) :- X < Y.
```

```
find_max3(X, Y, Z, X) :- X >= Y, X >= Z, !.
```

```
find_max3(X, Y, Z, Y) :- Y >= X, Y >= Z, !.
```

```
find_max3(X, Y, Z, Z) :- Z >= Y, Z >= X.
```

Output:

```
GNU Prolog console
File Edit Terminal Prolog Help
GNU Prolog 1.5.0 (64 bits)
Compiled Jul  8 2021, 12:22:53 with gcc
Copyright (C) 1999-2021 Daniel Diaz

| ?- [user].
compiling user for byte code...
find_max2(X, Y, X) :- X >= Y, !.
find_max2(X, Y, Y) :- X < Y.
find_max3(X, Y, Z, X) :- X >= Y, X >= Z, !.
find_max3(X, Y, Z, Y) :- Y >= X, Y >= Z, !.
find_max3(X, Y, Z, Z) :- Z >= Y, Z >= X.

user compiled, 5 lines read - 1452 bytes written, 3722 ms

yes
| ?- find_max2(2,3,W), write(W).
3

W = 3

yes
| ?- find_max3(2,3,4,W), write(W).
4

W = 4

yes
| ?- |
```

Experiment 7

Aim:

WAP to find factorial of a number.

Code:

factorial(0, 1).

factorial(N, M) :- N > 0, Prev is N -1, factorial(Prev, M1), M is M1*N.

Output:

```
GNU Prolog console
File Edit Terminal Prolog Help
GNU Prolog 1.5.0 (64 bits)
Compiled Jul  8 2021, 12:22:53 with gcc
Copyright (C) 1999-2021 Daniel Diaz

| ?- [user].
compiling user for byte code...
factorial(0, 1).
factorial(N, M) :- N > 0, Prev is N -1, factorial(Prev, M1), M is M1 * N.

user compiled, 2 lines read - 780 bytes written, 3120 ms

yes
| ?- factorial(5,W), write(W).
120

W = 120 ?

yes
| ?-
```


Experiment 8

Aim:

WAP to find sum of first N numbers.

Code:

sum(0,0).

sum(N,M) :- N > 0, Prev is N -1, sum(Prev,M1), M is M1 + N.

Output:

```
GNU Prolog console
File Edit Terminal Prolog Help
GNU Prolog 1.5.0 (64 bits)
Compiled Jul  8 2021, 12:22:53 with gcc
Copyright (C) 1999-2021 Daniel Diaz

| ?- [user].
compiling user for byte code...
sum(0,0).
sum(N,M) :- N > 0, Prev is N -1, sum(Prev,M1), M is M1 + N.

user compiled, 2 lines read - 750 bytes written, 2832 ms
yes
| ?- sum(5,W), write(W).
15

W = 15 ?

yes
| ?-
```

Experiment 9

Aim:

Write a program to find Fibonacci sequence upto Nth term.

Fibonacci Sequence is 0, 1, 1, 2, 3, 5, 8, ...

Code:

fibonacci(1,0).

fibonacci(2,1).

fibonacci(N,F) :- N>2, N1 is N-1, N2 is N-2, fibonacci(N1,F1), fibonacci(N2,F2), F is F1+F2.

Output:

```
GNU Prolog console
File Edit Terminal Prolog Help
GNU Prolog 1.5.0 (64 bits)
Compiled Jul  8 2021, 12:22:53 with gcc
Copyright (C) 1999-2021 Daniel Diaz

| ?- [user].
compiling user for byte code...
fibonacci(1,0).
fibonacci(2,1).
fibonacci(N,F) :- N>2, N1 is N-1, N2 is N-2, fibonacci(N1,F1), fibonacci(N2,F2), F is F1+F2.

user compiled, 3 lines read - 1100 bytes written, 13720 ms

(31 ms) yes
| ?- fibonacci(5,W), write(W).
3

W = 3 ?
yes
| ?- |
```

Experiment 10

Aim:

Write a program to represent following statements in Prolog.

- a. Prakash likes food if the food is edible and it tastes sweet.
- b. Chocolates tastes sweet.
- c. Toffees tastes sweet.
- d. Gourd tastes bitter.
- e. Chocolates are edible.
- f. Toffees are edible.
- g. Gourd is edible.

Further ask a question/query in Prolog such that the output of the question should be:

Chocolates -> Toffees -> No

Code:

```
likes(prakash, X) :- edible(X), tastes(X, sweet).
```

```
tastes(chocolates, sweet).
```

```
tastes(toffees, sweet).
```

```
tastes(gourd, bitter).
```

```
edible(chocolates).
```

```
edible(toffees).
```

```
edible(gourd).
```

Output:

```
GNU Prolog console
File Edit Terminal Prolog Help
GNU Prolog 1.5.0 (64 bits)
Compiled Jul  8 2021, 12:22:53 with gcc
Copyright (C) 1999-2021 Daniel Diaz

| ?- [user].
compiling user for byte code...
likes(prakash, X) :- edible(X), tastes(X, sweet).
tastes(chocolates, sweet).
tastes(toffees, sweet).
tastes(gourd, bitter).
edible(chocolates).
edible(toffees).
edible(gourd).

user compiled, 7 lines read - 998 bytes written, 10404 ms

(31 ms) yes
| ?- likes(prakash,X).

X = chocolates ? ;

X = toffees ? ;

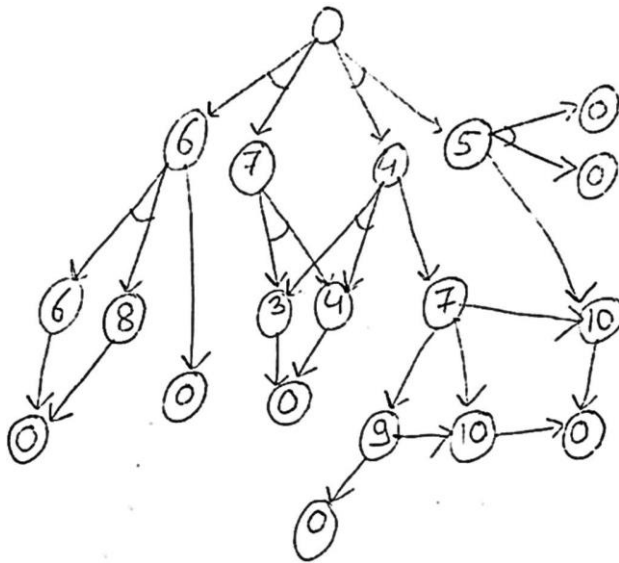
no
| ?- |
```

Experiment 11

Aim:

Write a program to solve AND OR Graph using AO* Search Algorithm.

Assume all edge weights are one. And the number inside every node represents the estimated time required to solve the sub problem at that node. The number 0 inside the node represents that sub problem is completely solved.



Code:

```
//Aneesh Panchal
//2K20/MC/21

#include<iostream>
#include<vector>
#include<queue>
#include<map>
#include<climits>
#include<algorithm>
#include<string>
using namespace std;

map<string,int> costCalc(map<char,int> &H, vector<string> &Condition, int weight=1){
    map<string,int> Updated_cost;
    for(auto i:Condition){
        Updated_cost[i] = 0;
        for(auto k:i){
            Updated_cost[i] += H[k] + weight;
        }
    }
    return Updated_cost;
}
```

```

map<char, map<string, int>> Updated_costs(map<char, int> &H, map<char, vector<string>>
&Conditions, int weight = 1){
    vector<char> Nodes;
    map<char, map<string, int>> UpdatedC;
    for(auto i:H){
        Nodes.push_back(i.first);
    }
    reverse(Nodes.begin(), Nodes.end());
    for(auto node:Nodes){
        if(Conditions.find(node) == Conditions.end()) continue;
        else{
            UpdatedC[node] = costCalc(H, Conditions[node], weight);
            int minC = INT_MAX;
            for(auto i:UpdatedC[node]){
                minC = min(minC, i.second);
            }
            H[node] = minC;
        }
    }
    return UpdatedC;
}

void printUpdatedC(map<char, map<string, int>> &Updated_cost){
    for(auto i:Updated_cost){
        cout<<i.first<<" -> { ";
        for(auto p:i.second){
            cout<<p.first<<": "<<p.second<<" ";
        }
        cout<<"}\n";
    }
}

void printShortestP(map<char, map<string, int>> &Updated_costs, char start){
    queue<char> Path;
    Path.push(start);
    while(!Path.empty()){
        string next;
        char curr;
        int v = INT_MAX;
        curr = Path.front();
        Path.pop();
        if(!Updated_costs[curr].empty()){
            for(auto i:Updated_costs[curr]){
                if(i.second < v){
                    v = i.second;
                    next = i.first;
                }
            }
        }
        if(v==INT_MAX){
            cout<<curr<<": 0\n";
        }
        else{

```

```

        for(auto i:next)
            Path.push(i);
        cout<<curr<<" -> "<<next[0];
        for(int i=1; i<next.size(); i++){
            cout<<" AND "<<next[i];
        }
        cout<<" : " <<v<<endl;
    }
}

int main(){
    map<char,int> HeuristicV
    {{'A',0},{'B',6},{'C',7},{'D',4},{'E',5},{'F',6},{'G',8},{'H',0},{'I',3},{'J',4},{'K',7},{
    'L',10},{'M',0},{'N',0},{'O',0},{'P',0},{'Q',9},{'R',10},{'S',0},{'T',0}};
    map<char,vector<string>>
    Conditions{{'A',{'BC',"DE"}},{'B',{'FG',"H"}},{'C',{'IJ'}},{'D',{'IJ',
    "K"}},{'E',{'MN',"L"}},{'F',{'O'}},{'G',{'O'}},{'I',{'P'}},{'J',{'P'}},{'K',{'Q',"R","L"}},
    {'L',{'T'}},{'Q',{'R',"S"}},{'R',{'T'}}}};
    cout<<endl;
    map<char,map<string,int>> UpdatedC = Updated_costs(HeuristicV,Conditions);
    printUpdatedC(UpdatedC);
    cout<<"\nShortest Path:\n";
    printShortestP(UpdatedC,'A');
    return 0;
}

```

Output:

```

PROBLEMS  OUTPUT  TERMINAL  SQL CONSOLE  DEBUG CONSOLE

PS E:\Codes\MC312> cd "e:\Codes\MC312\" ; if ($?) { g++ Exp11.cpp -o Exp11 } ; if ($?) { .\Exp11 }

A -> { BC:7 DE:7 }
B -> { FG:4 H:1 }
C -> { IJ:4 }
D -> { IJ:4 K:3 }
E -> { L:2 MN:2 }
F -> { O:1 }
G -> { O:1 }
I -> { P:1 }
J -> { P:1 }
K -> { L:2 Q:2 R:2 }
L -> { T:1 }
Q -> { R:2 S:1 }
R -> { T:1 }

Shortest Path:
A -> B AND C : 7
B -> H : 1
C -> I AND J : 4
H: 0
I -> P : 1
J -> P : 1
P: 0
P: 0

PS E:\Codes\MC312>

```