Kamla Nehru Mahavidyalaya,Nagpur
Department of MCA
Master in Computer Application[Semester-III]
Session 2024-25
Subject : Soft Computing
Execution List

1. Write an algorithm,Draw a flowchart and Write a program in CPP to implement Logic Gates.

```
#include <iostream>
using namespace std;
int main()
{ char menu; //Menu control variable
int result; //final output variable
int dataValue1;
int dataValue2;
cout<< "enter your Boolean operator code: (A,O,N,X): ";
cin>> menu;
switch (menu) //Menu control variable
{ case 'A':
        cout<< "Enter first Boolean value:";
        cin>> dataValue1;
        cout<< "Enter second Boolean value:";
        cin>> dataValue2;
        if(dataValue1 == 1 && dataValue2 == 1)
        {
        result = 1;
         }
        else
```

```
  {
result = 0;
  }
cout<< "show result:" << result;
break;
case 'O': cout<< "Enter first Boolean value:";
cin>> dataValue1;
cout<< "Enter second Boolean value:";
cin>> dataValue2;
if(dataValue1 == 1 || dataValue2 == 1)
  {
result = 1;
  }
else
  {
result = 0;
   }
        cout<< "show result:" << result;
        break;
        case 'N':
        cout<< "Enter first Boolean value:";
        cin>> dataValue1;
result = !dataValue1;
cout<< "show result:" << result;
break;
case 'X':
cout<< "Enter first Boolean value:";
```

```
cin>> dataValue1;
cout<< "Enter second Boolean value:";
cin>> dataValue2;
if(dataValue1 = !dataValue1)
{ result = 1; }
else { result = 0; }
cout<< "show result:" << result;
break;
default: result = 0;
break;
  }//end switch
cin.ignore(2);
return 0;
  }//end main
```

OUTPUT:

1) Enter your Boolean operator code :(A,O,N,X) : A
Enter First Boolean value :1
Enter Second Boolean value :1
Show Result :1

2) Enter your Boolean operator code :(A,O,N,X) : O
Enter First Boolean value :1
Enter Second Boolean value :0
Show Result :1

2. Write an algorithm,Draw a flowchart and Write a program in CPP to demonstrate the DFS Traversal on a Graph.

```
#include <iostream>
#include <list>
using namespace std;
//graph class for DFS travesal
class DFSGraph
{
int V;    // No. of vertices
list<int> *adjList;    // adjacency list
void DFS_util(int v, bool visited[]);  // A function used by DFS
public:
   // class Constructor
DFSGraph(int V)
   {
 this->V = V;
 adjList = new list<int>[V];
   }
   // function to add an edge to graph
void addEdge(int v, int w){
adjList[v].push_back(w); // Add w to v's list.
   }

void DFS();    // DFS traversal function
};
void DFSGraph::DFS_util(int v, bool visited[])
{
   // current node v is visited
visited[v] = true;
cout << v << " ";

   // recursively process all the adjacent vertices of the node
list<int>::iterator i;
for(i = adjList[v].begin(); i != adjList[v].end(); ++i)
if(!visited[*i])
DFS_util(*i, visited);
}
```

```
// DFS traversal
void DFSGraph::DFS()
{
   // initially none of the vertices are visited
bool *visited = new bool[V];
for (int i = 0; i < V; i++)
visited[i] = false;

   // explore the vertices one by one by recursively calling  DFS_util
for (int i = 0; i < V; i++)
if (visited[i] == false)
DFS_util(i, visited);
}

int main()
{
   // Create a graph
DFSGraph gdfs(5);
gdfs.addEdge(0, 1);
gdfs.addEdge(0, 2);
gdfs.addEdge(0, 3);
gdfs.addEdge(1, 2);
gdfs.addEdge(2, 4);
gdfs.addEdge(3, 3);
gdfs.addEdge(4, 4);

cout << "Depth-first traversal for the given graph:"<<endl;
gdfs.DFS();

return 0;
}
```

OUTPUT:-

Depth-first traversal for the given graph:-

```
    0   1 2 4 3
```

3. **Write an algorithm, Draw a flowchart and Write a program in CPP to demonstrate the BFS Traversal on a Graph.**

```
#include<iostream>
#include <list>

using namespace std;

// This class represents a directed graph using
// adjacency list representation
class Graph
{
   int V;    // No. of vertices

   // Pointer to an array containing adjacency
   // lists
   list<int> *adj;
public:
   Graph(int V);  // Constructor

   // function to add an edge to graph
   void addEdge(int v, int w);

   // prints BFS traversal from a given source s
   void BFS(int s);
};

Graph::Graph(int V)
{
   this->V = V;
   adj = new list<int>[V];
}

void Graph::addEdge(int v, int w)
{
   adj[v].push_back(w); // Add w to v's list.
}

void Graph::BFS(int s)
```

```
{
   // Mark all the vertices as not visited
   bool *visited = new bool[V];
   for(int i = 0; i < V; i++)
      visited[i] = false;

   // Create a queue for BFS
   list<int> queue;

   // Mark the current node as visited and enqueue it
   visited[s] = true;
   queue.push_back(s);

   // 'i' will be used to get all adjacent
   // vertices of a vertex
   list<int>::iterator i;

   while(!queue.empty())
   {
      // Dequeue a vertex from queue and print it
      s = queue.front();
      cout << s << " ";
      queue.pop_front();

      // Get all adjacent vertices of the dequeued
      // vertex s. If a adjacent has not been visited,
      // then mark it visited and enqueue it
      for (i = adj[s].begin(); i != adj[s].end(); ++i)
      {
         if (!visited[*i])
         {
            visited[*i] = true;
            queue.push_back(*i);
         }
      }
   }
}

// Driver program to test methods of graph class
int main()
```

```
{
    // Create a graph given in the above diagram
    Graph g(4);
    g.addEdge(0, 1);
    g.addEdge(0, 2);
    g.addEdge(1, 2);
    g.addEdge(2, 0);
    g.addEdge(2, 3);
    g.addEdge(3, 3);

    cout << "Following is Breadth First Traversal "
         << "(starting from vertex 2) \n";
    g.BFS(2);

    return 0;
}
```

OUTPUT :-

```
Following is Breadth First Traversal (starting from vertex 2)
2 0 3 1
--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

4. Write an algorithm,Draw a flowchart and Write a program in CPP to implement Perceptron Training Algorithm.

```
#include<iostream>
using namespace std;
int main()
{
int in[3],d,w[3],a=0;
for(int i=0;i<3;i++)
{
cout<<"\n initialize the weight vector w"<<i;
cin>>w[i];
 }
for(int i=0;i<3;i++)
{
cout<<"\n enter the input vector i"<<i;
cin>>in[i];
}
cout<<"\n enter the desined output";
cin>>d;
intans=1;
while(ans== 1)
{
for(int i=0;i<3;i++)
{
a = a + w[i] * in[i];
```

```
}

cout<<"\n desired output is"<<d;
cout<<"\n actual output is "<<a;
int e;
e=d-a;
cout<<"\n error is "<<e;
cout<<"\n press 1 to adjust weight else 0";
cin>>ans;
if (e<0)
{
for(int i=0;i<3;i++)
{
w[i]=w[i]-1;
}
}
else if(e>0)
{
for(int i=0;i<3;i++)
{
w[i]=w[i]+1;
}
}
}
}
```

OUTPUT:-

initialize the weight vector w0 3

initialize the weight vector w1 1

initialize the weight vector w2 7

enter the input vector i0 2

enter the input vector i1 3

enter the input vector i2 6

enter the desined output 9

desired output is 9

actual output is 51

error is -42

press 1 to adjust weight else 0

Desire output is 2

Actual output is 17

Error is -15

Press 1 to adjust weight else 0

5. Write an algorithm,Draw a flowchart and Write a program in CPP to implement Hebb's rule.

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

int main()
{
   int  m,n;
   cout<<"enter no.of features and no.of training datasets: \n";
   cin>> m>>n;

   int wt1[m], wt2[m];

   int input[n][m];
   cout<<"enter the input matrix row wise "<<endl;

   for(int i=0;i<n;i++)
   {
      for(int j=0;j<m;j++)
      {
         cin>>input[i][j];
      }
   }

   int target1[n], target2[n];

   cout<<" Enter the target in binary : "<<endl;
   for(int i=0;i<n;i++)
   {
      cin>>target1[i];
   }

   cout<<"Enter the target in bipolar: "<<endl;
   for(int i=0;i<n;i++)
   {
      cin>>target2[i];
   }
```

```
    for(int i=0;i<m;i++)
    {
       wt1[i]=0; //step 1: initialise all wts to 0
       wt2[i]=0;
    }

    for(int j=0;j<n;j++)
    {
       cout<<"###########j="<<j<<endl;
       for(int i=0; i<m;i++)
       {
          wt1[i]+= (input[j][i]*target1[j]);
          cout<<"weight1 at i="<<i<<" is "<<wt1[i]<<endl;
          wt2[i]+= (input[j][i]*target2[j]);
          cout<<"wt2 at i="<<i<<" is "<<wt2[i]<<endl;

       }

    }
    cout<<"************OUTPUT**************\nafter 1 epoch: binary
weights: "<<endl;

    for (int i = 0; i < m; ++i)
    {
       /* code */
       cout<<wt1[i]<<" ";
    }
    cout<<"\nafter 1 epoch: bipolar weights: "<<endl;

    for (int i = 0; i < m; ++i)
    {
       /* code */
       cout<<wt2[i]<<" ";
    }



    return 0;
}
```

OUTPUT:-

```
enter no.of features and no.of training datasets:
3
4
enter the input matrix row wise
-1 -1 1
-1 1 1
1 -1 1
1 1 1
 Enter the target in binary :
0
0
0
1
Enter the target in bipolar:
-1
-1
-1
1
###########j=0
weight1 at i=0 is 0
wt2 at i=0 is 1
weight1 at i=1 is 0
wt2 at i=1 is 1
weight1 at i=2 is 0
wt2 at i=2 is -1
###########j=1
weight1 at i=0 is 0
wt2 at i=0 is 2
weight1 at i=1 is 0
wt2 at i=1 is 0
weight1 at i=2 is 0
wt2 at i=2 is -2
###########j=2
weight1 at i=0 is 0
wt2 at i=0 is 1
weight1 at i=1 is 0
wt2 at i=1 is 1
```

```
weight1 at i=2 is 0
wt2 at i=2 is -3
############j=3
weight1 at i=0 is 1
wt2 at i=0 is 2
weight1 at i=1 is 1
wt2 at i=1 is 2
weight1 at i=2 is 1
wt2 at i=2 is -2
************OUTPUT**************
after 1 epoch: binary weights:
1 1 1
after 1 epoch: bipolar weights:
2 2 -2
--------------------------------
Process exited with return value 0
Press any key to continue . . .*/
```

6. Write an algorithm,Draw a flowchart and Write a program in CPP to implement ADALINE NETWORK.

```
#include<iostream>
#include<math.h>

using namespace std;
#define n 4
#define m 3
//n=no.of training sets
//m=no.of features
int main(int argc, char const *argv[])
{
   float w[]={0.1,0.1,0.1};
   float lr=0.1;
   int x[n][m]={{1,1,1}, {1,1,-1}, {1,-1,1},{1,-1,-1}};
   int target[]={1,1,1,-1};

   // for(int i=0;i<n;i++)
   //  {for(int j=0;j<m;j++)
   //     cout<<x[i][j]<<" ";
   //     cout<<endl;
   //  }

   float se[n];//squared error
   for(int i=0;i<n;i++)
   {
      cout<<"\n######### for training set i= "<<i;
      float y_in=0;
      for(int j=0;j<m;j++)
      {
         y_in+=w[j]*x[i][j];
      }
      cout<<"\nY_in = "<<y_in;


      float error= target[i]-y_in;
            cout<<"\nerror = "<<error;
```

```
        se[i]=error*error;
              cout<<"\nsquared error= "<<se[i];

        if(se[i]<lr)
        {
           cout<<"weights adjusted..training stopped";
           break;
        }
        else
        {
           for(int j=0;j<m;j++)
        {
              w[j]+=(lr*error*x[i][j]);
              cout<<"\nfor j= "<<j<< "weight is"<<w[j];

                    }
        }

    }
float mse=0;
    for(int k=0;k<n;k++)
    {
       mse+=se[k];

    }

    mse/=n;

    cout<<"\n\n************OUTPUT**************rmse =
"<<sqrt(mse)<<"\nweights:\n";

           for(int j=0;j<m;j++)
        {
cout<<w[j]<<" ";
                    }

    return 0;
}
```

OUTPUT :-

```
/*######### for training set i= 0
Y_in = 0.3
error = 0.7
squared error= 0.49
for j= 0weight is0.17
for j= 1weight is0.17
for j= 2weight is0.17
######### for training set i= 1
Y_in = 0.17
error = 0.83
squared error= 0.6889
for j= 0weight is0.253
for j= 1weight is0.253
for j= 2weight is0.087
######### for training set i= 2
Y_in = 0.087
error = 0.913
squared error= 0.833569
for j= 0weight is0.3443
for j= 1weight is0.1617
for j= 2weight is0.1783
######### for training set i= 3
Y_in = 0.0043
error = -1.0043
squared error= 1.00862
for j= 0weight is0.24387
for j= 1weight is0.26213
for j= 2weight is0.27873

**********OUTPUT**********rmse = 0.869064
weights:
0.24387 0.26213 0.27873
--------------------------------
Process exited with return value 0
Press any key to continue . . .*/
```

7. Write an algorithm,Draw a flowchart and Write a program in CPP for Error Back Propagation Algorithm(EBPA) Learning.

```
#include<iostream>
#include<math.h>
using namespace std;
int main()
{
float l,c,s1,n1,n2,w10,b10,w20,b20,w11,b11,w21,b21,p,t,a0=-1,a1,a2,e,s2;
cout<<"enter the input weights/base of second n/w= ";
cin>>w10>>b10;
cout<<"enter the input weights/base of second n/w= ";
cin>>w20>>b20;cout<<"enter the learning coefficient of n/w c= ";
cin>>c;/* Step1:Propagation of signal through n/w */
n1=w10*p+b10;a1=tanh(n1);
n2=w20*a1+b20;
a2=tanh(n2);
e=(t-a2);/* Back Propagation of Sensitivities */
s2=-2*(1-a2*a2)*e;
s1=(1-a1*a1)*w20*s2;/* Updation of weights and bases */
w21=w20-(c*s2*a1);
w11=w10-(c*s1*a0);
b21=b20-(c*s2);
b11=b10-(c*s1);
cout<<"The uploaded weight of first n/w w11= "<<w11;
cout<<"\n"<<"The uploaded weight of second n/w w21= "<<w21;
cout<<"\n"<<"The uploaded base of second n/w b11= "<<b11;
```

```
cout<<"\n"<<"The uploaded base of second n/w b21= "<<b21;
return 0;
}
```

OUTPUT:-

```
enter the input weights/base of second n/w= 12
35
enter the input weights/base of second n/w= 23
45
enter the learning coefficient of n/w c= 11
The uploaded weight of first n/w w11= 12
The uploaded weight of second n/w w21= 23
The uploaded base of second n/w b11= 35
The uploaded base of second n/w b21= 45
--------------------------------
Process exited after 29.3 seconds with return value 0
Press any key to continue . . .
```

8. Write an algorithm,Draw a flowchart and Write a program in CPP for SVM CLASSIFICATION

```
#include<iostream>
using namespace std;
int main( )
{
int in[3],d,w[3],a=0;
for(int i=0;i<3;i++)
{
cout<<"\n initialize the weight vector w"<<i;
cin>>w[i];
}
for(int i=0;i<3;i++)
{
cout<<"\n enter the input vector i"<<i;
cin>>in[i];
}
cout<<"\n enter the desired output";
cin>>d;
intans=1;
while(ans==1)
{
for(int i=0;i<3;i++)
{
a = a+w[i]* in[i];
}
cout<<"\n desired output is"<<d;
```

```
cout<<"\n actual output is "<<a;
int e;
e=d-a;
cout<<"\n error is"<<e;
cout<<"\n press 1 to adjust weight else 0";
cin>>ans;
if (e<0)
{
for(int i=0;i<3;i++)
{
w[i]=w[i]-1;
}
}
else if (e>0)
{
for(int i=0;i<3;i++)
{
w[i]=w[i]+1;
}
}
}
return 0;
}
```

OUTPUT:-

initialize the weight vector w0 2

initialize the weight vector w1 3

initialize the weight vector w2 4

enter the input vector i0 5

enter the input vector i1 6

enter the input vector i2 7

enter the desired output4

desired output is 4

actual output is 56

error is -52

press 1 to adjust weight else 0