D. Y. PATIL COLLEGE OF ENGINEERING

Akurdi, Pune – 411 044

LABORATORY Index

SUBJECT: Lab Practice I

INDEX

Sr. No.	Title of Experiment	Performance Date	Completion Date
1	Download the Iris flower dataset or any other dataset into a Data Frame. (eg https://archive.ics.uci.edu/ml/datasets/Iris) Use Python/R and Perform following – How many features are there and what are their types (e.g., numeric, nominal)? Compute and display summary statistics for each feature available in the dataset. (eg. minimum value, maximum value, mean, range, standard deviation, variance and percentiles Data Visualization-Create a histogram for each feature in the dataset to illustrate the feature distributions. Plot each histogram. Create a boxplot for each feature in the dataset. All of the boxplots should be combined into a single plot. Compare distributions and identify outliers	3-08-2021	5-08-2021
2	Download Pima Indians Diabetes dataset. Use Naive Bayes" Algorithm for classification Load the data from CSV file and split it into training and test datasets. summarize the properties in the training dataset so that we can calculate probabilities and make predictions. Classify samples from a test dataset and a summarized training dataset.	5-08-2021	10-08-2021
3	Bigmart Sales Analysis: For data comprising of transaction records of a sales store. The data has 8523 rows of 12 variables. **Predict the sales of a store.** Sample Test data set available here https://datahack.analyticsvidhya.com/contest/practice-problem-big-mart-sales-iii/	10-08-2021	12-08-2021
4	Twitter Data Analysis: Use Twitter data for sentiment analysis. The dataset is 3MB in size and has 31,962 tweets. Identify the tweets which are hate tweets and which are not	12-08-2021	17-08-2021
5	**MINI Project Based On Data Analytic**	17-10-2021	24-12-2021
6	Implement Parallel Reduction using Min, Max, Sum and Average operations.	24-08-2021	26-08-2021
7	Vector and Matrix Operations Design parallel algorithm to 1. Add two large vectors 2. Multiply Vector and Matrix 3. Multiply two N × N arrays using n2 processors	26-08-2021	31-08-2021
8	Parallel Sorting Algorithms For Bubble Sort and Merger Sort, based on existing sequential algorithms, design and implement parallel algorithm utilizing all resources available.	31-08-2021	2-09-2021
9	Parallel Search Algorithm Design and implement parallel algorithm utilizing all resources available. for Binary Search for Sorted Array Depth-First Search (tree or an undirected graph) OR Breadth-First Search (tree or an undirected graph) OR Best-First Search that (traversal of graph to reach a target in the shortest possible path)	2-09-2021	7-09-2021
10	**Mini Project Based On High performance computing**	21-10-2021	29-12-2021

D. Y. PATIL COLLEGE OF ENGINEERING

Akurdi, Pune – 411 044

LABORATORY Index

SUBJECT: Lab Practice I

11	Implement Tic-Tac-Toe using A* algorithm	7-09-2021	13-09-2021
12	Solve 8-puzzle problem using A* algorithm. Assume any initial configuration and define goal configuration clearly	13-09-2021	15-09-2021
13	Develop elementary chatbot for suggesting investment as per the customers need.	15-09-2021	21-09-2021
14	Implement goal stack planning for the following configurations from the blocks world/Monkeys-Banana Problem	21-09-2021	22-09-2021
15	**Mini Project Based On Robotics**	22-10-2021	29-12-2021
	Beyond Syllabus		
16	Virtual Lab Assignment Fuzzy Logic	13-11-2021	13-11-2021

Mr. Suraj V Kurde
Lab Teacher

Dr. M A Potey
HoD, Computer Engineering

Name: Vikas Mane
Class: BE-B
Roll Number: B1921152
PRN Number: 72000291F

Assignment 1

Title: Download the Iris flower dataset or any other dataset into a Data Frame. (e.g., https://archive.ics.uci.edu/ml/datasets/Iris) Use Python/R and Perform following –

- How many features are there and what are their types (e.g., numeric, nominal)?
- Compute and display summary statistics for each feature available in the dataset. (e.g. minimum value, maximum value, mean, range, standard deviation, variance and percentiles
- Data Visualization-Create a histogram for each feature in the dataset to illustrate the feature distributions. Plot each histogram.
- Create a boxplot for each feature in the dataset. All of the boxplots should be combined into a single plot. Compare distributions and identify outliers.

Aim: Implement a dataset into a data frame. Implement the following operations:
1. Display data set details.
2. Calculate min, max, mean, range, standard deviation, variance.
3. Create histograph using hist function.
4. Create boxplot using boxplot function.

System Requirements: Python, Data Visualization libraries, Jupyter Notebook.

Theory:

Data analytics (DA): Data analytics (DA) is the *process of examining data sets in order to draw conclusions about the information they contain, increasingly with the aid of specialized systems and software.* Data analytics technologies and techniques are widely used in commercial industries to enable organizations to make more-informed business decisions and by scientists and researchers to verify or disprove scientific models, theories and hypotheses.

Data Science: Dealing with unstructured and structured data, Data Science is a field that comprises of everything that related to *data cleansing, preparation, and analysis.*

Data Science is the combination of statistics, mathematics, programming, problem-solving, capturing data in ingenious ways, the ability to look at things differently, and the activity of cleansing, preparing and aligning the data.

In simple terms, it is the umbrella of techniques used when trying to extract insights and information from data.

Big Data: *Big Data refers to humongous volumes of data that cannot be processed effectively with the traditional applications that exist.* The processing of Big Data begins with the raw data that isn't aggregated and is most often impossible to store in the memory of a single computer.

A buzzword that is used to describe immense volumes of data, both unstructured and structured, Big Data inundates a business on a day-to-day basis. Big Data is something that can be used to analyze insights which can lead to better decisions and strategic business moves.

The definition of Big Data, given by Gartner is, “Big data is high-volume, and high-velocity and/or high-variety information assets that demand cost-effective, innovative forms of information processing

Data Analytics: Data Analytics the science of examining raw data with the purpose of drawing conclusions about that information.

Data Analytics involves applying an algorithmic or mechanical process to derive insights. For example, running through a number of data sets to look for meaningful correlations between each other.

It is used in a number of industries to allow the organizations and companies to make better decisions as well as verify and disprove existing theories or models.

The focus of Data Analytics lies in inference, which is the process of deriving conclusions that are solely based on what the researcher already knows.

Applications of Data Analysis:

- Healthcare: The main challenge for hospitals with cost pressures tightens is to treat as many patients as they can efficiently, keeping in mind the improvement of the quality of care. Instrument and machine data is being used increasingly to track as well as optimize patient flow, treatment, and equipment used in the hospitals. It is estimated that there will be a 1% efficiency gain that could yield more than $63 billion in the global healthcare savings.
- Travel: Data analytics is able to optimize the buying experience through the mobile/ weblog and the social media data analysis. Travel sights can gain insights into the customer’s desires and preferences. Products can be up-sold by correlating the current sales to the subsequent browsing increase browse-to-buy conversions via customized packages and offers. Personalized travel recommendations can also be delivered by data analytics based on social media data.
- Gaming: Data Analytics helps in collecting data to optimize and spend within as well as across games. Game companies gain insight into the dislikes, the relationships, and the likes of the users.
- Energy Management: Most firms are using data analytics for energy management, including smart-grid management, energy optimization, energy distribution, and building automation in utility companies. The application here is centered on the controlling and monitoring of network devices, dispatch crews, and manage service outages. Utilities are given the ability to integrate millions of data points in the network performance and lets the engineers use the analytics to monitor the network.

Objective: To learn the concept of how to display summary statistics for each feature available in the dataset

Outcome: After completion of this assignment, it is expected to implement various statistics.

Input: Structured Dataset: Iris Dataset

Output:

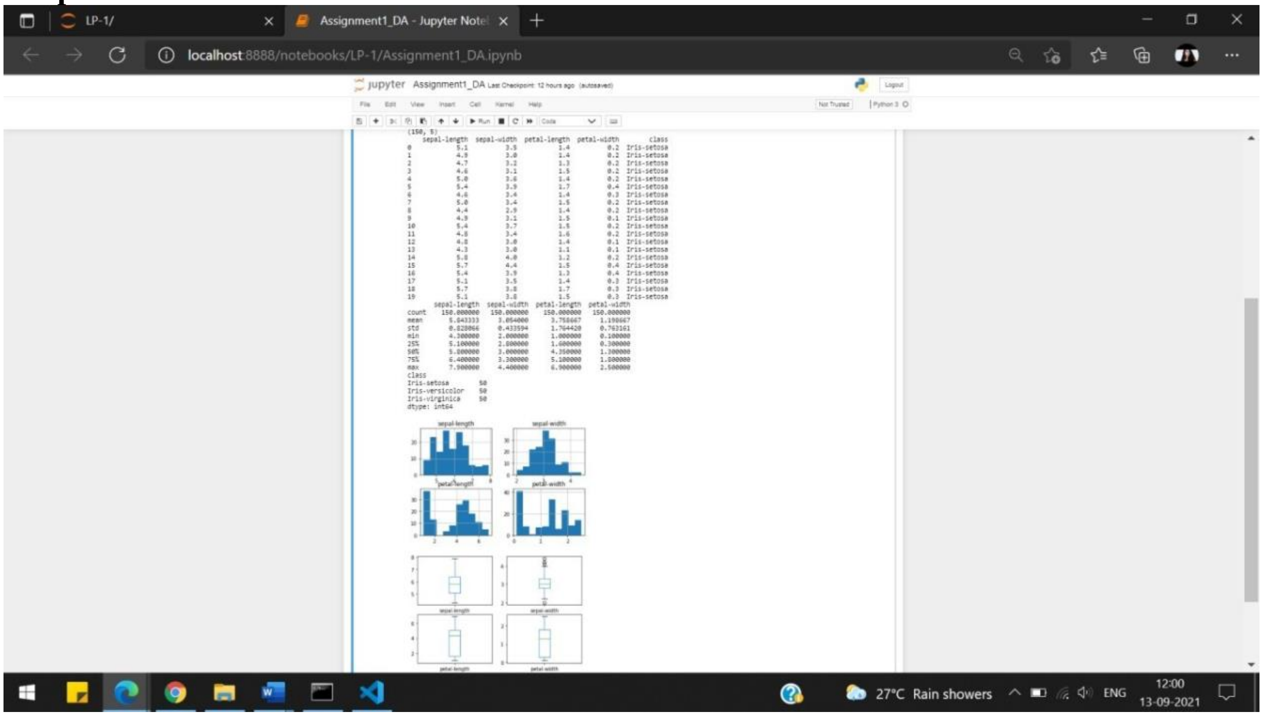
1. Dataset Details.
2. Min, Max, Mean, Variance value and Percentiles of probabilities 3 Histogram using Hist Function.
4. Boxplot using Boxplot Function.

Result / Conclusion: We learned to analyse data distribution, and data statistics for given dataset.

Program:

```
# Load libraries
import pandas
import matplotlib.pyplot as plt

# Load dataset
url = "https://archive.ics.uci.edu/ml/machine-learning-
databases/iris/iris.data"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width',
'class']
dataset = pandas.read_csv(url, names=names)

# shape
print(dataset.shape)

# head
print(dataset.head(20))

# descriptions
print(dataset.describe())

# class distribution
print(dataset.groupby('class').size())


# histograms
dataset.hist()
plt.show()

# box and whisker plots
dataset.plot(kind='box', subplots=True, layout=(2,2), sharex=False,
sharey=False)
plt.show()
```

Output:



```
(150, 5)
    sepal-length  sepal-width  petal-length  petal-width        class
0            5.1          3.5           1.4          0.2  Iris-setosa
1            4.9          3.0           1.4          0.2  Iris-setosa
2            4.7          3.2           1.3          0.2  Iris-setosa
3            4.6          3.1           1.5          0.2  Iris-setosa
4            5.0          3.6           1.4          0.2  Iris-setosa
5            5.4          3.9           1.7          0.4  Iris-setosa
6            4.6          3.4           1.4          0.3  Iris-setosa
7            5.0          3.4           1.5          0.2  Iris-setosa
8            4.4          2.9           1.4          0.2  Iris-setosa
9            4.9          3.1           1.5          0.1  Iris-setosa
10           5.4          3.7           1.5          0.2  Iris-setosa
11           4.8          3.4           1.6          0.2  Iris-setosa
12           4.8          3.0           1.4          0.1  Iris-setosa
13           4.3          3.0           1.1          0.1  Iris-setosa
14           5.8          4.0           1.2          0.2  Iris-setosa
15           5.7          4.4           1.5          0.4  Iris-setosa
16           5.4          3.9           1.3          0.4  Iris-setosa
17           5.1          3.5           1.4          0.3  Iris-setosa
18           5.7          3.8           1.7          0.3  Iris-setosa
19           5.1          3.8           1.5          0.3  Iris-setosa
       sepal-length  sepal-width  petal-length  petal-width
count    150.000000   150.000000    150.000000   150.000000
mean       5.843333     3.054000      3.758667     1.198667
std        0.828066     0.433594      1.764420     0.763161
min        4.300000     2.000000      1.000000     0.100000
25%        5.100000     2.800000      1.600000     0.300000
50%        5.800000     3.000000      4.350000     1.300000
75%        6.400000     3.300000      5.100000     1.800000
max        7.900000     4.400000      6.900000     2.500000
class
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
dtype: int64
```

localhost:8888/notebooks/LP-1/Assignment1_DA.ipynb
Assignment1_DA - Jupyter Notebook
jupyter Assignment1_DA Last Checkpoint: 12 hours ago (autosaved)
File Edit View Insert Cell Kernel Help
Run
Code
sepal-length
sepal-width
petal-length
petal-width
27°C Rain showers
ENG
12:00
13-09-2021

Name: Vikas Mane

Class: BE-B

Roll Number: B1921152

PRN Number: 72000291F

ASSIGNMENT 2

Problem Statement : Download Pima Indians Diabetes dataset. Use Naive Bayes‟ Algorithm for classification

- Load the data from CSV file and split it into training and test datasets.
- summarize the properties in the training dataset so that we can calculate probabilities and make predictions.
- Classify samples from a test dataset and a summarized training dataset.

System Requirements: Python, Data Visualization libraries, Jupyter Notebook.

Theory :

The test problem we will use in this tutorial is the Pima Indians Diabetes problem.

This problem is comprised of 768 observations of medical details for Pima indians patents. The records describe instantaneous measurements taken from the patient such as their age, the number of times pregnant and blood workup. All patients are women aged 21 or older. All attributes are numeric, and their units vary from attribute to attribute.

Each record has a class value that indicates whether the patient suffered an onset of diabetes within 5 years of when the measurements were taken (1) or not (0).

This is a standard dataset that has been studied a lot in machine learning literature. A good prediction accuracy is 70%-76%.

Datasets for training, testing and validation of Algorithms :

In machine learning, the study and construction of algorithms that can learn from and make predictions on data is a common task. Such algorithms work by making data-driven predictions or decisions, through building a mathematical model from input data.

The data used to build the final model usually comes from multiple datasets. In particular, three data sets are commonly used in different stages of the creation of the model.

The model is initially fit on a **training dataset**, that is a set of examples used to fit the parameters (e.g. weights of connections between neurons in artificial neural networks) of the model. The model (e.g. a neural net or a naive Bayes classifier) is trained on the training dataset using a supervised learning method (e.g. gradient descent or stochastic gradient descent). In practice, the training dataset often consist of pairs of an input vector and the corresponding *answer* vector or scalar, which is commonly denoted as the *target*. The current model is run with the training dataset and produces a result, which is then compared with the *target*, for each input vector in the training dataset. Based on the result of the comparison and the specific learning algorithm being used, the parameters of the model are adjusted. The model fitting can include both variable selection and parameter estimation.

Successively, the fitted model is used to predict the responses for the observations in a second dataset called the **validation dataset**. The validation dataset provides an unbiased evaluation of a model fit on the training dataset while tuning the model's hyperparameters (e.g. the number of hidden units in a neural network). Validation datasets can be used for regularization by early stopping: stop training when the error on the validation dataset increases, as this is a sign of overfitting to the training dataset. This simple procedure is complicated in practice by the fact that the validation dataset's error may fluctuate during training, producing multiple local minima. This complication has led to the creation of many ad-hoc rules for deciding when overfitting has truly begun.

Finally, the **test dataset** is a dataset used to provide an unbiased evaluation of a *final* model fit on the training dataset.

NAIVE BAYES

What is Naive Bayes Classifier?

Have you ever wondered how your mail provider implements spam filtering or how online news channels perform news text classification or even how companies perform sentiment analysis of their audience on social media? All of this and more are done through a machine learning algorithm called Naive Bayes Classifier.

Naive Bayes
Named after Thomas Bayes from the 1700s who first coined this in the Western literature. Naive Bayes classifier works on the principle of conditional probability as given by the Bayes theorem.

Advantages of Naive Bayes Classifier
Listed below are six benefits of Naive Bayes Classifier.

- Very simple and easy to implement
- Needs less training data
- Handles both continuous and discrete data
- Highly scalable with the number of predictors and data points

- As it is fast, it can be used in real-time predictions
- Not sensitive to irrelevant features

Bayes Theorem

We will understand Bayes Theorem in detail from the points mentioned below.

- According to the Bayes model, the conditional probability P(Y|X) can be calculated as: P(Y|X) = P(X|Y)P(Y) / P(X)
- This means you have to estimate a very large number of P(X|Y) probabilities for a relatively small vector space X.
- For example, for a Boolean Y and 30 possible Boolean attributes in the X vector, you will have to estimate 3 billion probabilities P(X|Y).
- To make it practical, a Naïve Bayes classifier is used, which assumes conditional independence of P(X) to each other, with a given value of Y.
- This reduces the number of probability estimates to 2*30=60 in the above example.

Algorithm Steps to follow :

Algorithm is divided in following steps :

Handle Data: Load the data from CSV file and split it into training and test datasets.
Summarize Data: summarize the properties in the training dataset so that we can calculate probabilities and make predictions.
Make a Prediction: Use the summaries of the dataset to generate a single prediction.

1. Handle Data

The first thing we need to do is load our data file. The data is in CSV format without a header line or any quotes. We can open the file with the open function and read the data lines using the reader function in the csv module.

We also need to convert the attributes that were loaded as strings into numbers that we can work with them.

Next we need to split the data into a training dataset that Naive Bayes can use to make predictions and a test dataset that we can use to evaluate the accuracy of the model. We need to split the data set randomly into train and datasets with a ratio of 67% train and 33% test (this is a common ratio for testing an algorithm on a dataset).

2. Summarize Data

The naive bayes model is comprised of a summary of the data in the training dataset. This summary is then used when making predictions.

The summary of the training data collected involves the mean and the standard deviation for each attribute, by class value. For example, if there are two class values and 7 numerical attributes, then we need a mean and standard deviation for each attribute (7) and class value (2) combination, that is 14 attribute summaries.

These are required when making predictions to calculate the probability of specific attribute values belonging to each class value.

We can break the preparation of this summary data down into the following sub-tasks:

1. Separate Data By Class

2. Calculate Mean
3. Calculate Standard Deviation
4. Summarize Dataset
5. Summarize Attributes By Class

Separate Data By Class

The first task is to separate the training dataset instances by class value so that we can calculate statistics for each class. We can do that by creating a map of each class value to a list of instances that belong to that class and sort the entire dataset of instances into the appropriate lists.

Calculate Mean

We need to calculate the mean of each attribute for a class value. The mean is the central middle or central tendency of the data, and we will use it as the middle of our gaussian distribution when calculating probabilities.

We also need to calculate the standard deviation of each attribute for a class value. The standard deviation describes the variation of spread of the data, and we will use it to characterize the expected spread of each attribute in our Gaussian distribution when calculating probabilities.

The standard deviation is calculated as the square root of the variance. The variance is calculated as the average of the squared differences for each attribute value from the mean. Note we are using the N-1 method, which subtracts 1 from the number of attribute values when calculating the variance.

Summarize Dataset

Now we have the tools to summarize a dataset. For a given list of instances (for a class value) we can calculate the mean and the standard deviation for each attribute.

The zip function groups the values for each attribute across our data instances into their own lists so that we can compute the mean and standard deviation values for the attribute.

Summarize Attributes By Class

We can pull it all together by first separating our training dataset into instances grouped by class. Then calculate the summaries for each attribute.

3. Make Prediction

We are now ready to make predictions using the summaries prepared from our training data.
Making predictions involves calculating the probability that a given data instance belongs to each class, then selecting the class with the largest probability as the prediction.
We can divide this part into the following tasks:

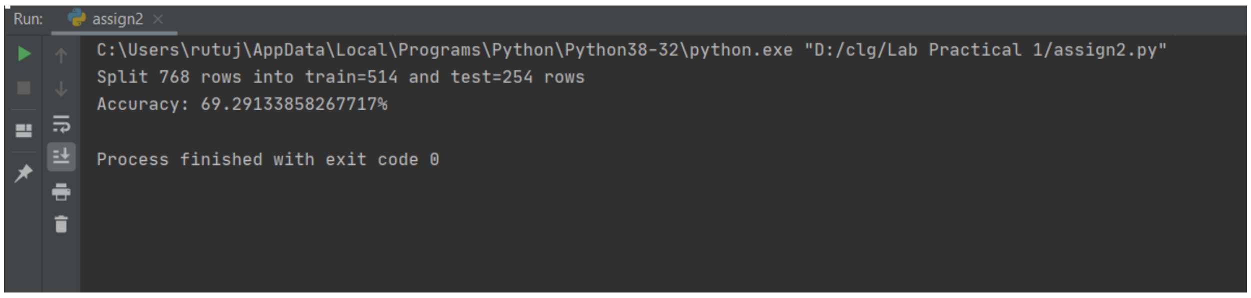
1. Calculate Gaussian Probability Density Function
2. Calculate Class Probabilities
3. Make a Prediction
4. Estimate Accuracy

INPUT:

```
import csv
import random
import math

def loadCsv(filename):
	lines = csv.reader(open(filename, "rb"))
	dataset = list(lines)
```

```
    for i in range(len(dataset)):
        dataset[i] = [float(x) for x in dataset[i]]
    return dataset

def splitDataset(dataset, splitRatio):
    trainSize = int(len(dataset) * splitRatio)
    trainSet = []
    copy = list(dataset)
    while len(trainSet) < trainSize:
        index = random.randrange(len(copy))
        trainSet.append(copy.pop(index))
    return [trainSet, copy]

def separateByClass(dataset):
    separated = {}
    for i in range(len(dataset)):
        vector = dataset[i]
        if (vector[-1] not in separated):
            separated[vector[-1]] = []
        separated[vector[-1]].append(vector)
    return separated

def mean(numbers):
    return sum(numbers)/float(len(numbers))

def stdev(numbers):
    avg = mean(numbers)
    variance = sum([pow(x-avg,2) for x in numbers])/float(len(numbers)-1)
    return math.sqrt(variance)

def summarize(dataset):
    summaries = [(mean(attribute), stdev(attribute)) for attribute in zip(*dataset)]
    del summaries[-1]
    return summaries

def summarizeByClass(dataset):
    separated = separateByClass(dataset)
    summaries = {}
    for classValue, instances in separated.iteritems():
        summaries[classValue] = summarize(instances)
    return summaries

def calculateProbability(x, mean, stdev):
    exponent = math.exp(-(math.pow(x-mean,2)/(2*math.pow(stdev,2))))
    return (1 / (math.sqrt(2*math.pi) * stdev)) * exponent

def calculateClassProbabilities(summaries, inputVector):
	probabilities = {}
	for classValue, classSummaries in summaries.iteritems():
		probabilities[classValue] = 1
		for i in range(len(classSummaries)):
			mean, stdev = classSummaries[i]
			x = inputVector[i]
			probabilities[classValue] *= calculateProbability(x, mean, stdev)
	return probabilities

def predict(summaries, inputVector):
	probabilities = calculateClassProbabilities(summaries, inputVector)
	bestLabel, bestProb = None, -1
	for classValue, probability in probabilities.iteritems():
		if bestLabel is None or probability > bestProb:
			bestProb = probability
			bestLabel = classValue
	return bestLabel

def getPredictions(summaries, testSet):
	predictions = []
	for i in range(len(testSet)):
		result = predict(summaries, testSet[i])
		predictions.append(result)
	return predictions

def getAccuracy(testSet, predictions):
	correct = 0
	for i in range(len(testSet)):
		if testSet[i][-1] == predictions[i]:
			correct += 1
	return (correct/float(len(testSet))) * 100.0

def main():
	filename = 'pima-indians-diabetes.data.csv'
	splitRatio = 0.67
	dataset = loadCsv(filename)
	trainingSet, testSet = splitDataset(dataset, splitRatio)
	print('Split {0} rows into train={1} and test={2} rows').format(len(dataset),
len(trainingSet), len(testSet))
	# prepare model
	summaries = summarizeByClass(trainingSet)
	# test model
	predictions = getPredictions(summaries, testSet)
	accuracy = getAccuracy(testSet, predictions)
	print('Accuracy: {0}%').format(accuracy)

main()
```

Output:



Run: assign2

```
C:\Users\rutuj\AppData\Local\Programs\Python\Python38-32\python.exe "D:/clg/Lab Practical 1/assign2.py"
Split 768 rows into train=514 and test=254 rows
Accuracy: 69.29133858267717%

Process finished with exit code 0
```

Conclusion :

We learned to classification of data based on Bayes' probability theorem.

Name: Vikas Mane

Roll No: B1921152

Class: BE-B

Assignment 3

<u>Title:</u> Big-Mart Sales Analysis: For data comprising of transaction records of a sales store.

<u>Aim:</u> Predict the sales of a store. The data has 8523 rows of 12 variables.

<u>System Requirements:</u> Python, Data Visualization Libraries, Jupyter Notebook.

<u>Theory:</u>

- In today’s modern world, huge shopping centers such as big malls and marts are recording data related to sales of items or products with their various dependent or independent factors as an important step to be helpful in prediction of future demands and inventory management. The dataset built with various dependent and independent variables is a composite form of item attributes, data gathered by means of customer, and also data related to inventory management in a data warehouse. The data is thereafter refined in order to get accurate predictions and gather new as well as interesting results that shed a new light on our knowledge with respect to the task’s data. This can then further be used for forecasting future sales by means of employing machine learning algorithms such as the random forests and simple or multiple linear regression model.
- BigMart’s data scientists collected sales data of their 10 stores situated at different locations with each store having 1559 different products as per 2013 data collection. Using all the observations it is inferred what role certain properties of an item play and how they affect their sales.
- On implementation, the prediction results show the correlation among different attributes considered and how a particular location of medium size recorded the highest sales, suggesting that other shopping locations should follow similar patterns for improved sales. Multiple instances parameters and various factors can be used to make this sales prediction more innovative and successful. Accuracy, which plays a key role in prediction-based systems, can be significantly increased as the number of parameters used are increased. Also, a look into how the sub-models work can lead to increase in productivity of system

<u>Objective:</u>

To analyze the sales of a store

<u>Input:</u>

Structured Dataset : Big Mart Sales

<u>Output:</u>

1. Histograms / Bar Graphs for Analysis
2. Mean Squared Error using Bayesian Linear Regression

<u>Result / Conclusion:</u>

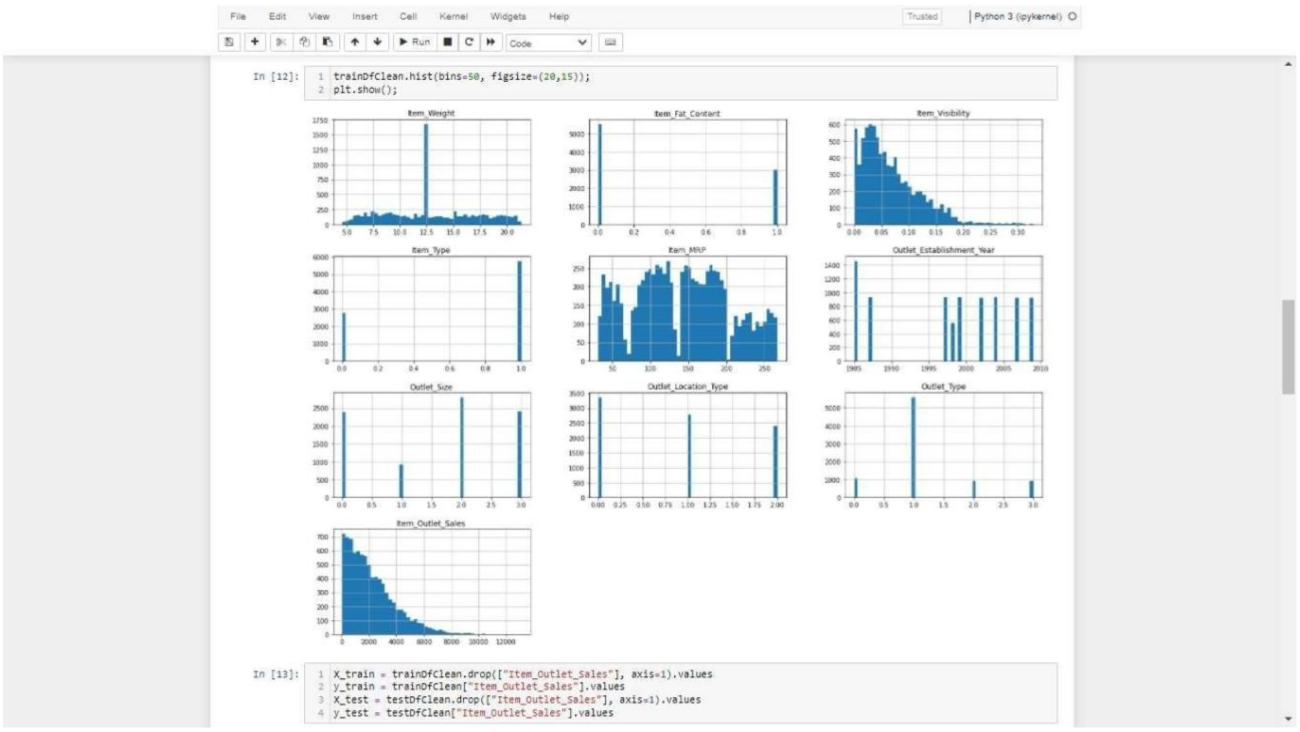
Hence, we have completed the analysis of Big Mart Store.

Program:

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from numpy import linalg

trainDf = pd.read_csv("train.csv")
testDf = pd.read_csv("test.csv")
trainingSetIndex = len(trainDf)

trainDf.head()

trainDf.dtypes

print(trainDf.isnull().sum())

combination = trainDf.append(testDf)
combination = combination.drop(["Item_Identifier", "Outlet_Identifier"], axis=1)

# replacing the null in the ItemWeight
combination = combination.fillna(combination.median())

# replacing nominal values
combination["Item_Fat_Content"] = combination["Item_Fat_Content"].replace({"LF": 0,
"reg": 1})
combination["Item_Fat_Content"] = combination["Item_Fat_Content"].replace({"Low
Fat": 0, "Regular": 1})
combination["Item_Fat_Content"] = combination["Item_Fat_Content"].replace({"low
fat": 0, "Regular": 1})

perishable = ["Breads", "Breakfast", "Dairy", "Fruits and Vegetables", "Meat",
"Seafood"]
non_perishable = ["Baking Goods", "Canned", "Frozen Foods", "Hard Drinks", "Health
and Hygiene", "Household", "Soft Drinks", "Snack Foods", "Starchy Foods", "Others"]

combination["Item_Type"] = combination["Item_Type"].replace(to_replace=perishable,
value="perishable")
combination["Item_Type"] =
combination["Item_Type"].replace(to_replace=non_perishable, value="non_perishable")
combination["Item_Type"] = combination["Item_Type"].replace({"perishable": 0,
"non_perishable": 1})

combination["Outlet_Size"] = combination["Outlet_Size"].replace({"Small": 0,
"High": 1, "Medium": 2, np.nan: 3})
combination["Outlet_Location_Type"] =
combination["Outlet_Location_Type"].replace({"Tier 3": 0, "Tier 2": 1, "Tier 1":
2})

combination["Outlet_Type"] = combination["Outlet_Type"].replace({"Grocery Store":
0,"Supermarket Type1": 1, "Supermarket Type2": 2, "Supermarket Type3": 3})

# splitting again the cleaned data sets
trainDfClean = combination[:trainingSetIndex]
testDfClean = combination[trainingSetIndex:]

trainDfClean.head()

testDfClean.head()

trainDfClean.hist(bins=50, figsize=(20,15));
plt.show();

X_train = trainDfClean.drop(["Item_Outlet_Sales"], axis=1).values
```

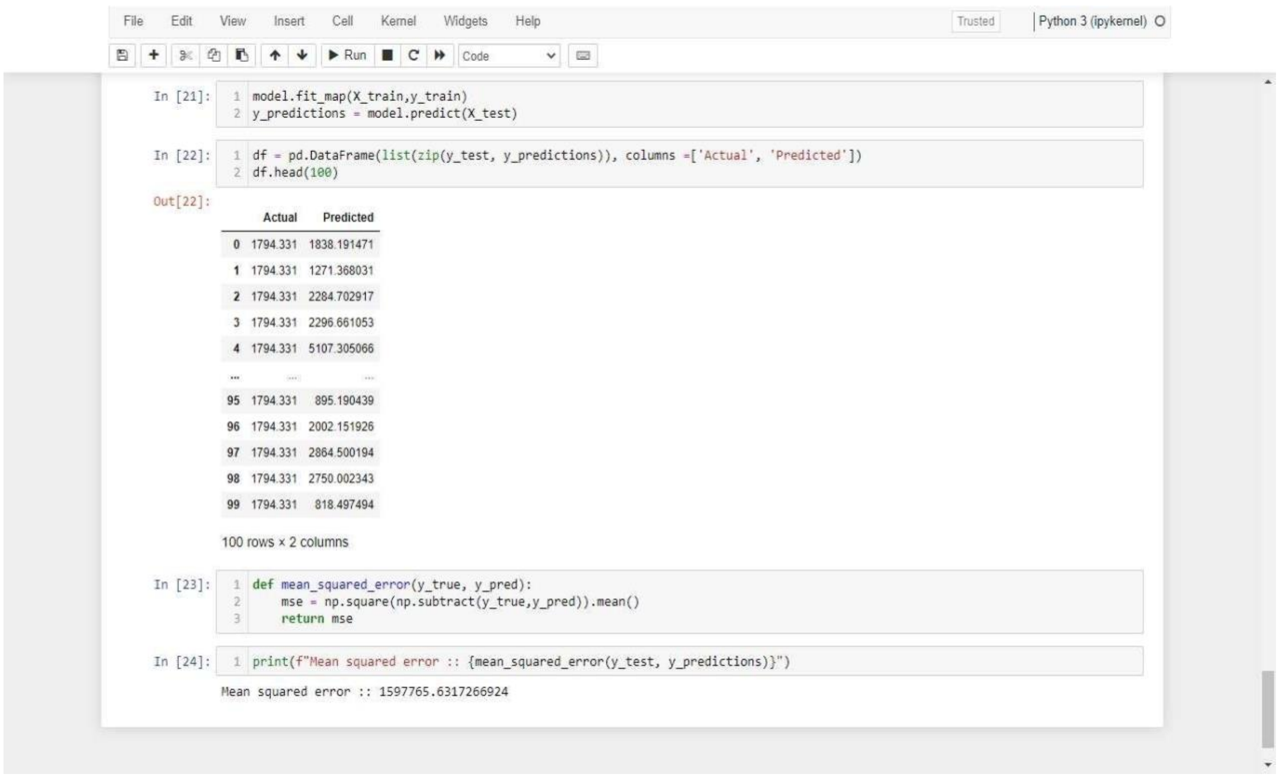
```
y_train = trainDfClean["Item_Outlet_Sales"].values
X_test = testDfClean.drop(["Item_Outlet_Sales"], axis=1).values
y_test = testDfClean["Item_Outlet_Sales"].values

from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
lreg = LinearRegression();
lreg.fit(X_train,y_train);
y_pred = lreg.predict(X_test);

mean_absolute_error(y_test, y_pred)

mean_squared_error(y_test, y_pred)

df = pd.DataFrame(list(zip(y_test, y_pred)), columns =['Actual', 'Predicted'])
df.head(100)

def add_intercept(X):
    X_new = np.ones((X.shape[0], X.shape[1] + 1), dtype=X.dtype)
    X_new[:, 1:] = X[:, :]
    return X_new


class BayesianLinearRegression:
    """
    Linear regression model: y = z beta[1] + beta[0]
    beta ~ N(0,Lambda)
    Lambda = I * lambda
    P(y|x,beta) ~ N(y|x.dot(beta),sigma**2)
    """

    def___init_(self, lamb=20., beta_mu=0, sigma=5, fit_intercept=True):
        """
        lamb: variance of the prior for each of the feature dimensions.
        beta_mu: the mean of the prior
        sigma: variance of the prediction error.
        """
        if not np.isscalar(lamb):
            self.inv_lamb = 1. / np.asarray(lamb)
        else:
            self.inv_lamb = 1. / float(lamb)
        if not np.isscalar(beta_mu):
            self.beta_mu = np.asarray(beta_mu)
        else:
            self.beta_mu = float(beta_mu)

        self.sigma = sigma
        self.fit_intercept = fit_intercept
        self.beta = None

    def fit_ml(self, X, y):
        """
          Fit a Maximum Likelihood estimate. (not Bayesian)
          X: features, n_samples by n_features nd-array
          y: target values, n_samples array
        """
        if  self.fit_intercept:
            X = add_intercept(X)
        self.beta = linalg.inv(X.T.dot(X)).dot(X.T.dot(y))

    def fit_map(self, X, y):
        """
          Fit a MAP estimate
          X: features, n_samples by n_features nd-array
          y: target values, n_samples array
        """
        if  self.fit_intercept:
            X = add_intercept(X)
        # data  setup
        f_dim = X.shape[1]
        if np.isscalar(self.inv_lamb):
            inv_lamb = np.diagflat(np.repeat(self.inv_lamb, f_dim))
        else:
            inv_lamb = np.diagflat(self.inv_lamb)
        if np.isscalar(self.beta_mu):
            beta_mu = np.repeat(self.beta_mu, f_dim)
        else:
            beta_mu = self.beta_mu
        sigma = self.sigma
        # let the actual calculation begin
        l = sigma ** 2 * inv_lamb
        s = linalg.inv(X.T.dot(X) + l)
        # adding in the mean of the prior
        b0 = sigma ** 2 * inv_lamb.dot(beta_mu)

        self.beta = s.dot(X.T.dot(y) + b0)

    def predict(self, X):
        """ Prediction """
        if  self.fit_intercept:
            X = add_intercept(X)
        return X.dot(self.beta)

model = BayesianLinearRegression()

X_train = X_train.astype(float)
X_test = X_test.astype(float)

model.fit_map(X_train,y_train)
y_predictions = model.predict(X_test)

df = pd.DataFrame(list(zip(y_test,y_predictions)),columns =['Actual', 'Predicted'])
df.head(100)

def mean_squared_error(y_true, y_pred):
    mse = np.square(np.subtract(y_true,y_pred)).mean()
    return mse

print(f"Mean squared error :: {mean_squared_error(y_test, y_predictions)}")
```

Output:



```
In [12]: trainDfClean.hist(bins=50, figsize=(20,15));
         plt.show();
```

```
In [13]: X_train = trainDfClean.drop(["Item_Outlet_Sales"], axis=1).values
         y_train = trainDfClean["Item_Outlet_Sales"].values
         X_test = testDfClean.drop(["Item_Outlet_Sales"], axis=1).values
         y_test = testDfClean["Item_Outlet_Sales"].values
```



```
In [21]: model.fit_map(X_train,y_train)
         y_predictions = model.predict(X_test)
```

```
In [22]: df = pd.DataFrame(list(zip(y_test, y_predictions)), columns =['Actual', 'Predicted'])
         df.head(100)
```

Out[22]:

	Actual	Predicted
0	1794.331	1838.191471
1	1794.331	1271.368031
2	1794.331	2284.702917
3	1794.331	2296.661053
4	1794.331	5107.305066
...	...	...
95	1794.331	895.190439
96	1794.331	2002.151926
97	1794.331	2864.500194
98	1794.331	2750.002343
99	1794.331	818.497494

100 rows × 2 columns

```
In [23]: def mean_squared_error(y_true, y_pred):
             mse = np.square(np.subtract(y_true,y_pred)).mean()
             return mse
```

```
In [24]: print(f"Mean squared error :: {mean_squared_error(y_test, y_predictions)}")

Mean squared error :: 1597765.6317266924
```

NAME : VIKAS MANE

PRN : 72000291F

Roll No: B1921152

Class: BE-B

Assignment 4

<u>Title:</u> Twitter Data Analysis: Use Twitter data for sentiment analysis. The dataset is 3MB in size and has 31,962 tweets. Identify the tweets which are hate tweets and which are not

<u>Aim:</u> Implement the sentiment analysis of twitter data. Implement the following operations:
1. Sentiment analysis of twitter dataset.
2. Classify the tweets as positive and negative tweets from dataset

<u>System Requirements:</u> Python, Data Visualization libraries, Jupyter Notebook.

<u>Theory:</u>

Python Regular Expression Library: Regular expressions are used to identify whether a pattern exists in a given sequence of characters (string) or not. They help in manipulating textual data, which is often a pre-requisite for data science projects that involve text mining. You must have come across some application of regular expressions: they are used at the server side to validate the format of email addresses or password during registration, used for parsing text data files to find, replace or delete certain string, etc.

Python Tweepy Library: This library provides access to the entire twitter RESTful API methods. Each method can accept various parameters and return responses.

Python TextBlob Library: TextBlob is a Python library for processing textual data. It provides a consistent API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, and more.

Authentication: Create OAuthHandler object : Tweepy supports oauth authentication. Authentication is handled by the tweepy AuthHandler class.

Utility Function in Python : This function provides sentiments analysis of tweets

<u>Objective:</u>

To learn the concept of how the data set analysis is done by taking example of Twitter Sentiment Analysis

<u>Outcome:</u>

After completion of this assignment, it is expected to understand data analysis using various features and implementation of Twitter Sentiment Analysis.

<u>Input:</u>

Structured Dataset : Twitter Dataset
Dataset File: Twitter.csv

<u>Output:</u>

1. Sentiment analysis of twitter dataset.
2. Categorization of tweets as positive and negative tweets.

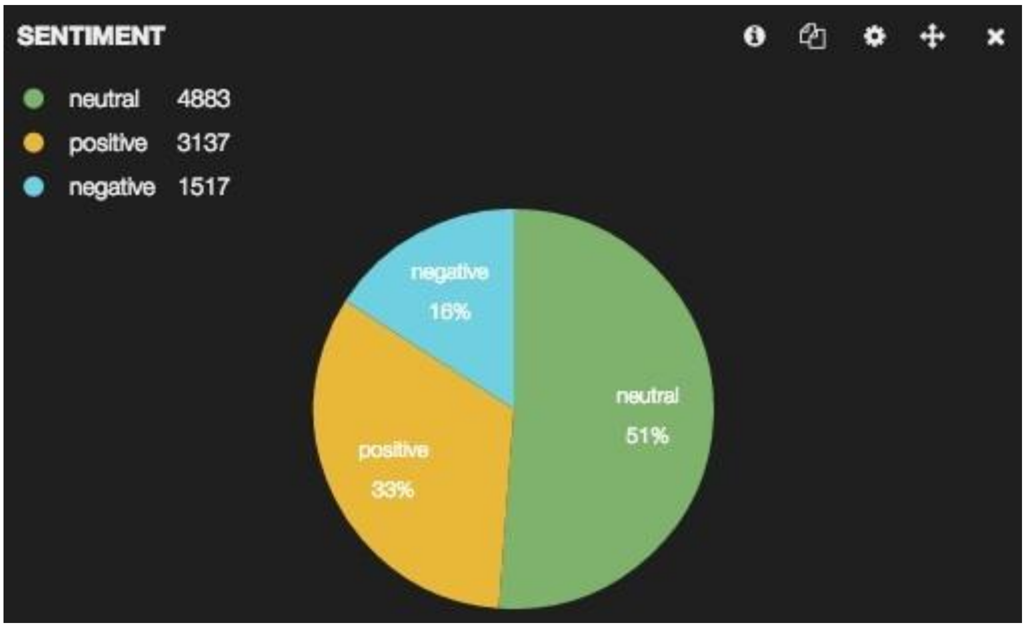
<u>Result / Conclusion:</u>

Hence, we have studied sentiment analysis of Twitter dataset to classify the tweets from dataset.

Program:

```
import re
import tweepy
from tweepy import OAuthHandler
from textblob import TextBlob

class TwitterClient(object):
      '''
      Generic Twitter Class for sentiment analysis.
      '''
      def___init__(self):
            '''
            Class constructor or initialization method.
            '''
            # keys and tokens from the Twitter Dev Console
            consumer_key = 'XXXXXXXXXXXXXXXXXXXXXXXX'
            consumer_secret = 'XXXXXXXXXXXXXXXXXXXXXXXXXXXX'
            access_token = 'XXXXXXXXXXXXXXXXXXXXXXXXXXX'
            access_token_secret = 'XXXXXXXXXXXXXXXXXXXXXXXXX'

            # attempt authentication
            try:
                  # create OAuthHandler object
                  self.auth = OAuthHandler(consumer_key, consumer_secret)
                  # set access token and secret
                  self.auth.set_access_token(access_token, access_token_secret)
                  # create tweepy API object to fetch tweets
                  self.api = tweepy.API(self.auth)
            except:
                  print("Error: Authentication Failed")

      def clean_tweet(self, tweet):
            '''
            Utility function to clean tweet text by removing links, special
characters
            using simple regex statements.
            '''
            return ' '.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])
                                                |(\w+:\/\/\S+)", " ",
tweet).split())

      def get_tweet_sentiment(self, tweet):
            '''
            Utility function to classify sentiment of passed tweet
            using textblob's sentiment method
            '''
            # create TextBlob object of passed tweet text
            analysis = TextBlob(self.clean_tweet(tweet))
            # set sentiment
            if analysis.sentiment.polarity > 0:
                  return 'positive'
            elif analysis.sentiment.polarity == 0:
                  return 'neutral'
            else:
                  return 'negative'

      def get_tweets(self, query, count = 10):
            '''
            Main function to fetch tweets and parse them.
            '''
            # empty list to store parsed tweets
            tweets = []

            try:
                  # call twitter api to fetch tweets
                  fetched_tweets = self.api.search(q = query, count = count)

                  # parsing tweets one by one
                  for tweet in fetched_tweets:
                        # empty dictionary to store required params of a tweet
                        parsed_tweet = {}

                        # saving text of tweet
                        parsed_tweet['text'] = tweet.text
                        # saving sentiment of tweet
                        parsed_tweet['sentiment'] =
self.get_tweet_sentiment(tweet.text)

                        # appending parsed tweet to tweets list
                        if tweet.retweet_count > 0:
                              # if tweet has retweets, ensure that it is
appended only once
                              if parsed_tweet not in tweets:
                                    tweets.append(parsed_tweet)
                        else:
                              tweets.append(parsed_tweet)

                  # return parsed tweets
                  return tweets

            except tweepy.TweepError as e:
                  # print error (if any)
                  print("Error : " + str(e))

def main():
      # creating object of TwitterClient Class
      api = TwitterClient()
      # calling function to get tweets
      tweets = api.get_tweets(query = 'Donald Trump', count = 200)

      # picking positive tweets from tweets
      ptweets = [tweet for tweet in tweets if tweet['sentiment'] ==
'positive']
      # percentage of positive tweets
      print("Positive tweets percentage: {}
%".format(100*len(ptweets)/len(tweets)))
      # picking negative tweets from tweets
      ntweets = [tweet for tweet in tweets if tweet['sentiment'] ==
'negative']
      # percentage of negative tweets
      print("Negative tweets percentage: {}
%".format(100*len(ntweets)/len(tweets)))
      # percentage of neutral tweets
      print("Neutral tweets percentage: {} % \
            ".format(100*(len(tweets) -(len( ntweets )+len(
ptweets)))/len(tweets)))

      # printing first 5 positive tweets
      print("\n\nPositive tweets:")
      for tweet in ptweets[:10]:
            print(tweet['text'])

      # printing first 5 negative tweets
      print("\n\nNegative tweets:")
      for tweet in ntweets[:10]:
            print(tweet['text'])

if___name___== "__main__":
      # calling main function
      main()
```

Output:



Name: Vikas Mane

Class: BE-B

Roll Number: B1921152

PRN Number: 72000291F

Assignment No.5

Title: a) Implement Parallel Reduction using Min, Max, Sum and Average operations.
b) Write a CUDA program that, given an N-element vector, find-

- The maximum element in the vector
- The minimum element in the vector
- The arithmetic mean of the vector
- The standard deviation of the values in the vector

Test for input N and generate a randomized vector V of length N (N should be large). The program should generate output as the two computed maximum values as well as the time taken to find each value.

Objective: To study and implementation of directive based parallel programming model. To study and implement the operations on vector, generate o/p as two computed max values as well as time taken to find each value

Outcome: Students will understand the implementation of sequential program augmented with compiler directives to specify parallelism. Students will understand the implementation of operations on vector, generate o/p as two computed max with respect to time.

Pre-requisites: 64-bit Open source Linux or its derivative, Programming Languages: C/C++,CUDA Tutorials.

Hardware Specification: x86_64 bit, 2 – 2/4 GB DDR RAM, 80 - 500 GB SATA HD, 1GB NIDIA TITAN X Graphics Card.

Software Specification: Ubuntu 14.04, GPU Driver 352.68, CUDA Toolkit 8.0, CUDNN Library v5.0

Theory:

Introduction:

OpenMP:

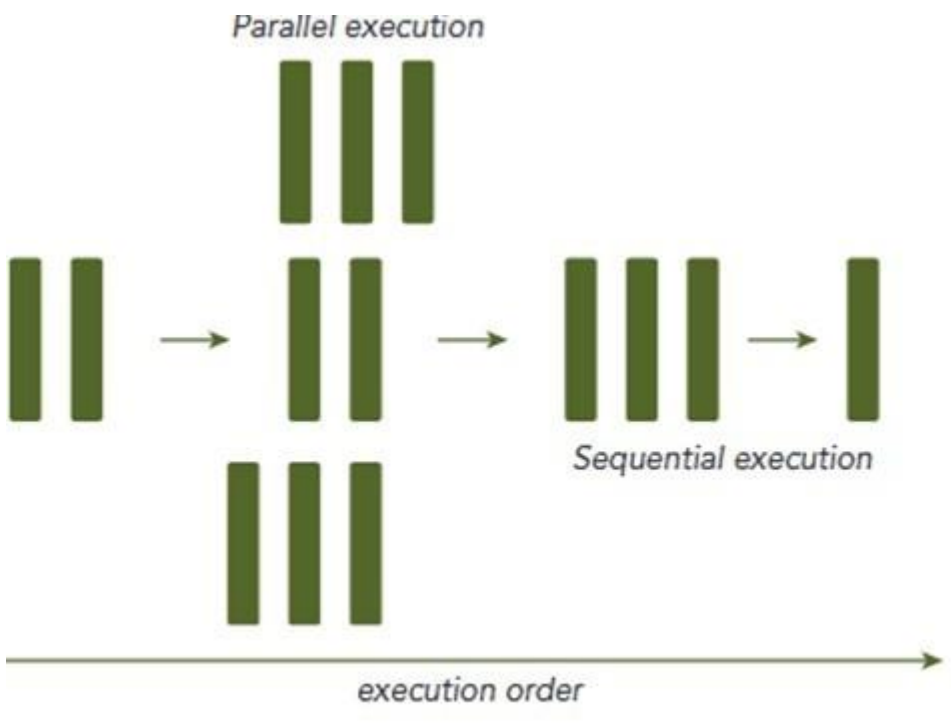
OpenMP is a set of C/C++ pragmas (or FORTRAN equivalents) which provide the programmer a high-level front-end interface which get translated as calls to threads (or other similar entities). The key phrase here is "higher-level"; the goal is to better enable the programmer to "think parallel," alleviating him/her of the burden and distraction of dealing with setting up and coordinating threads. For example, the OpenMP directive.

Parallel Programming:

There are two fundamental types of parallelism in applications:

- ➤Task parallelism
- ➤Data parallelism

Task parallelism arises when there are many tasks or functions that can be operated independently and largely in parallel. Task parallelism focuses on distributing functions across multiple cores.
Data parallelism arises when there are many data items that can be operated on at the same time. Data parallelism focuses on distributing the data across multiple cores.



CUDA:

CUDA programming is especially well-suited to address problems that can be expressed as data-parallel computations. Any applications that process large data sets can use a data- parallel model to speed up the computations. Data-parallel processing maps data elements to parallel threads. The first step in designing a data parallel program is to partition data across threads, with each thread working on a portion of the data.

CUDA Architecture:

A heterogeneous application consists of two parts:

➤Host code

➤Device code

Host code runs on CPUs and device code runs on GPUs. An application executing on a heterogeneous platform is typically initialized by the CPU. The CPU code is responsible for managing the environment, code, and data for the device before loading compute-intensive tasks on the device. With computational intensive applications, program sections often exhibit a rich amount of data parallelism. GPUs are used to accelerate the execution of this portion of data parallelism. When a hardware component that is physically separate from the CPU is used to accelerate computationally intensive sections of an application, it is referred to as a hardware accelerator. GPUs are arguably the most common example of a hardware accelerator. GPUs must operate in conjunction with a CPU-based host through a PCI-Express bus, as shown in Figure.

NVIDIA's CUDA nvcc compiler separates the device code from the host code during the compilation process. The device code is written using CUDA C extended with keywords for labeling data-parallel functions, called kernels . The device code is further compiled by Nvcc. During the link stage, CUDA runtime libraries are added for kernel procedure calls and explicit GPU device manipulation. Further kernel function, named helloFromGPU, to print the string of "Hello World from

GPU!" as follows:

```
__global__void helloFromGPU(void)
{
printf("Hello World from GPU!\n");
}
```

The qualifier __global_tells the compiler that the function will be called from the CPU and exe- cuted on the GPU. Launch the kernel function with the following code:

```
helloFromGPU <<<1,10>>>();
```

Triple angle brackets mark a call from the host thread to the code on the device side. A kernel is executed by an array of threads and all threads run the same code. The parameters within the triple angle brackets are the execution configuration, which specifies how many threads will execute the kernel. In this example, you will run 10 GPU threads.

A typical processing flow of a CUDA program follows this pattern:

1. Copy data from CPU memory to GPU memory.
2. Invoke kernels to operate on the data stored in GPU memory.
3. Copy data back from GPU memory to CPU memory.

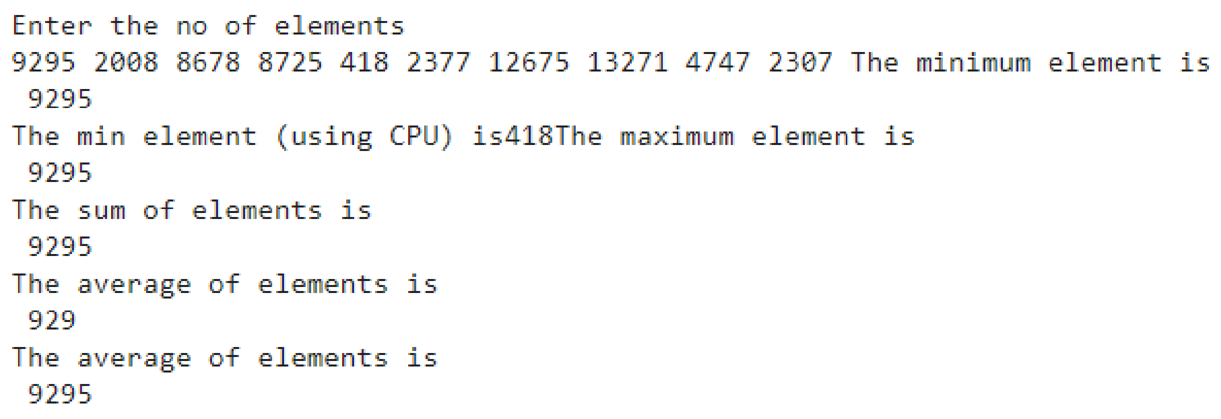
Input:

```
#include<iostream>
#include<cstdio>
#include<cstdlib>
#include<cuda_runtime.h>
using namespace std;


__global__void minimum(int *input)
{
       int tid=threadIdx.x;
       auto step_size=1;
  int number_of_threads=blockDim.x;

  while(number_of_threads>0)
  {
      if(tid<number_of_threads)
      {
          int first=tid*step_size*2;
          int second=first+step_size;
          if(input[second]<input[first])
            input[first]=input[second];
      }
      step_size=step_size*2;
      number_of_threads/=2;
  }

}
```

```
__global__void max(int *input)
{
   int tid=threadIdx.x;
   auto step_size=1;
   int number_of_threads=blockDim.x;

   while(number_of_threads>0)
   {
       if(tid<number_of_threads)
       {
           int first=tid*step_size*2;
           int second=first+step_size;
           if(input[second]>input[first])
            input[first]=input[second];
       }
       step_size*=2;
       number_of_threads/=2;
   }
}

__global__void sum(int *input)
{
    const int tid=threadIdx.x;
    auto  step_size=1;
    int number_of_threads=blockDim.x;
    while(number_of_threads>0)
    {
        if(tid<number_of_threads)
        {
            const int first=tid*step_size*2;
            const int second=first+step_size;
            input[first]=input[first]+input[second];
        }
    step_size = step_size*2;;
             number_of_threads =number_of_threads/2;
    }
}

__global__void average(int *input) //You can use above sum() to calculate sum and
divide it by num_of_elememts
{
    const int tid=threadIdx.x;
    auto  step_size=1;
    int number_of_threads=blockDim.x;
    int totalElements=number_of_threads*2;
    while(number_of_threads>0)
    {
        if(tid<number_of_threads)
        {
            const int first=tid*step_size*2;
            const int second=first+step_size;
            input[first]=input[first]+input[second];
        }
        step_size = step_size*2;;
```

```
              number_of_threads =number_of_threads/2;
     }
     input[0]=input[0]/totalElements;
}

int main()
{

       cout<<"Enter the no of elements"<<endl;
       int n;
       n=10;
  srand(n);
       int *arr=new int[n];
  int min=20000;
   //# Generate Input array using rand()
       for(int i=0;i<n;i++)
       {
              arr[i]=rand()%20000;
      if(arr[i]<min)
        min=arr[i];
    cout<<arr[i]<<" ";
       }

       int size=n*sizeof(int); //calculate no. of bytes for array
       int *arr_d,result1;

  //# Allocate memory for min Operation
       cudaMalloc(&arr_d,size);
       cudaMemcpy(arr_d,arr,size,cudaMemcpyHostToDevice);

  minimum<<<1,n/2>>>(arr_d);

       cudaMemcpy(&result1,arr_d,sizeof(int),cudaMemcpyDeviceToHost);

       cout<<"The minimum element is \n "<<result1<<endl;

  cout<<"The min element (using CPU) is"<<min;


  //#MAX OPERATION
  int *arr_max,maxValue;
  cudaMalloc(&arr_max,size);
       cudaMemcpy(arr_max,arr,size,cudaMemcpyHostToDevice);

  max<<<1,n/2>>>(arr_max);

       cudaMemcpy(&maxValue,arr_max,sizeof(int),cudaMemcpyDeviceToHost);

       cout<<"The maximum element is \n "<<maxValue<<endl;

  //#SUM OPERATION
  int *arr_sum,sumValue;
  cudaMalloc(&arr_sum,size);
       cudaMemcpy(arr_sum,arr,size,cudaMemcpyHostToDevice);
  sum<<<1,n/2>>>(arr_sum);

       cudaMemcpy(&sumValue,arr_sum,sizeof(int),cudaMemcpyDeviceToHost);

       cout<<"The sum of elements is \n "<<sumValue<<endl;

  cout<<"The average of elements is \n "<<(sumValue/n)<<endl;

  //# OR-----------

  //#AVG OPERATION
  int *arr_avg,avgValue;
  cudaMalloc(&arr_avg,size);
       cudaMemcpy(arr_avg,arr,size,cudaMemcpyHostToDevice);

  average<<<1,n/2>>>(arr_avg);

       cudaMemcpy(&avgValue,arr_avg,sizeof(int),cudaMemcpyDeviceToHost);

       cout<<"The average of elements is \n "<<avgValue<<endl;


  //# Free all allcated device memeory
   cudaFree(arr_d);
   cudaFree(arr_sum);
   cudaFree(arr_max);
   cudaFree(arr_avg);




return 0;

}
```

Output:



```
Enter the no of elements
9295 2008 8678 8725 418 2377 12675 13271 4747 2307 The minimum element is
 9295
The min element (using CPU) is418The maximum element is
 9295
The sum of elements is
 9295
The average of elements is
 929
The average of elements is
 9295
```

Conclusion: We have implemented parallel reduction using Min, Max, Sum and Average Operations. We have implemented CUDA program for calculating Min, Max, Arithmetic mean and Standard deviation Operations on N-element vector.

Name: Vikas Mane

Class: BE-B

Roll Number: B1921152

PRN Number: 72000291F

Experiment No: 6

Title: Vector and Matrix Operations-

Design parallel algorithm to

1. Add two large vectors
2. Multiply Vector and Matrix
3. Multiply two N × N arrays using n2 processors

Aim: Implement *nxn* matrix parallel addition, multiplication using CUDA, use shared memory.

Prerequisites:

- Concept of matrix addition, multiplication.

- Basics of CUDA programming

Objectives:

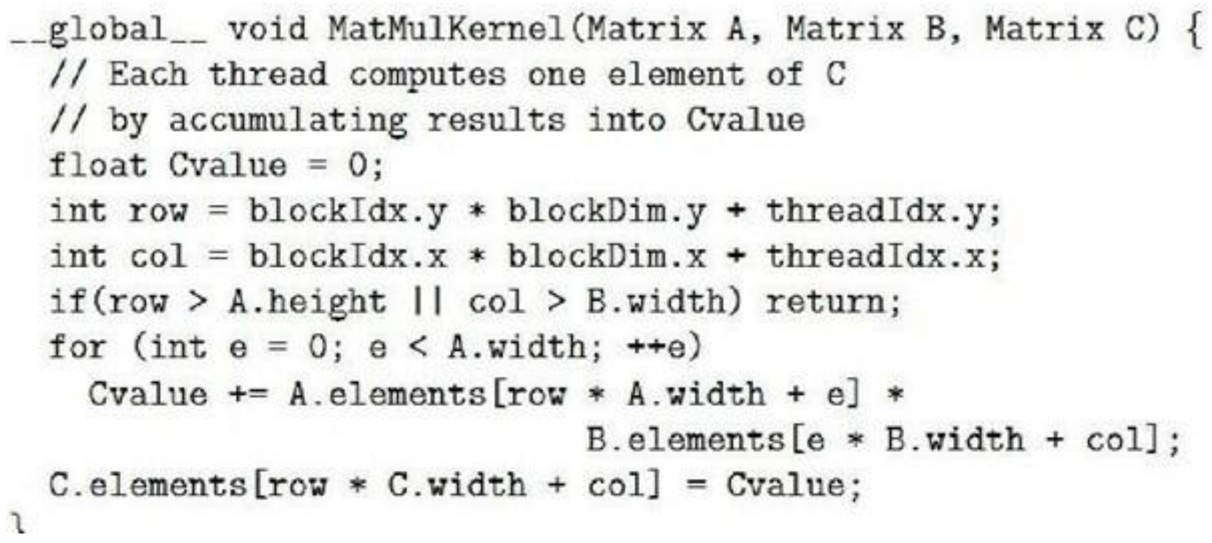
Student should be able to learn parallel programming, CUDA architecture and CUDA processing flow

Theory:

It has become increasingly common to see supercomputing applications harness the massive parallelism of graphics cards (Graphics Processing Units or GPUs) to speed up computations. One platform for doing so is NVIDIA's Compute Unified Device Architecture (CUDA). We use the example of Matrix Multiplication to introduce the basics of GPU computing in the CUDA environment.

Matrix Multiplication is a fundamental building block for scientific computing. Moreover, the algorithmic patterns of matrix multiplication are representative. Many other algorithms share similar optimization techniques as matrix multiplication. Therefore, matrix multiplication is one of the most important examples in learning parallel programming.

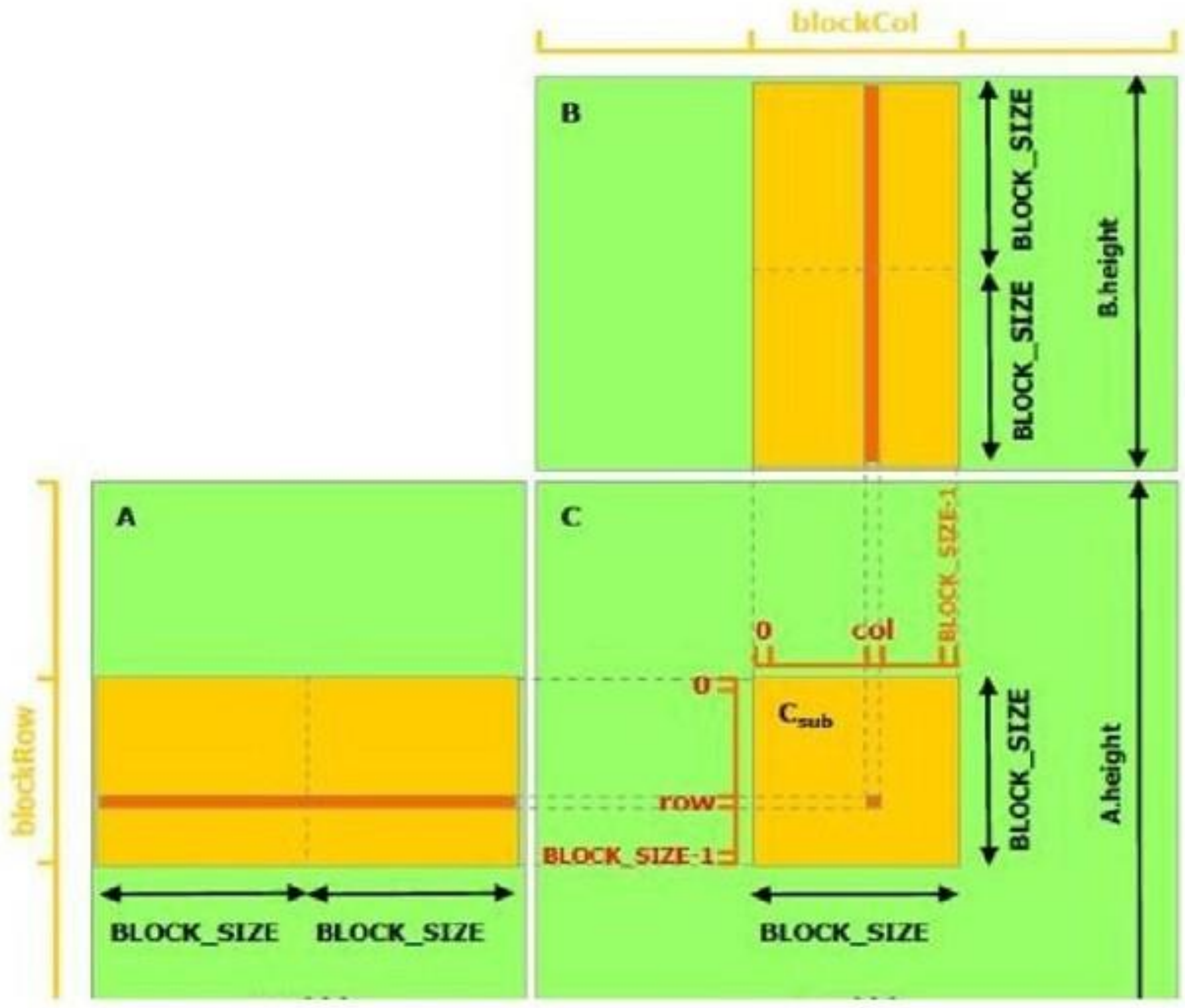
A kernel that allows host code to offload matrix multiplication to the GPU. The kernel function is shown below,



```
__global__ void MatMulKernel(Matrix A, Matrix B, Matrix C) {
  // Each thread computes one element of C
  // by accumulating results into Cvalue
  float Cvalue = 0;
  int row = blockIdx.y * blockDim.y + threadIdx.y;
  int col = blockIdx.x * blockDim.x + threadIdx.x;
  if(row > A.height || col > B.width) return;
  for (int e = 0; e < A.width; ++e)
    Cvalue += A.elements[row * A.width + e] *
                             B.elements[e * B.width + col];
  C.elements[row * C.width + col] = Cvalue;
}
```

The first line contains the___**global**_keyword declaring that this is an entry-point function for running code on the device. The declaration float **Cvalue = 0** sets aside a register to hold this float value where we will accumulate the product of the row and column entries. The next two lines help the thread to discover its row and column within the matrix. It is a good idea to make sure you understand those two lines before moving on. The ***if*** statement in the next line terminates the thread if its row or column place it outside the bounds of the product matrix. This will happen only in those blocks that overhang either the right or bottom side of the matrix.
The next three lines loop over the entries of the row of **A** and the column of **B** (these have the same size) needed to compute the (row, col)-entry of the product, and the sum of these products are accumulated in the **Cvalue** variable. Matrices **A** and **B** are stored in the device's global memory in row major order, meaning that the matrix is stored as a one-dimensional array, with the first row followed by the second row, and so on. Thus to find the index in this linear array of the **(i, j)** entry of matrix **A.** Finally, the last line of the kernel copies this product into the appropriate element of the product matrix C, in the device's global memory.

In light of the memory hierarchy described above, each threads loads **(2 × A. width)**
elements in Kernel .



From global memory — two for each iteration through the loop, one from matrix **A** and one from matrix **B**. Since accesses to global memory are relatively slow, this can bog down the kernel, leaving the threads idle for hundreds of clock cycles, for each access.
Matrix **A** is shown on the left and matrix **B** is shown at the top, with matrix **C**, their product, on the bottom-right. This is a nice way to lay out the matrices visually, since each element of **C** is the product of the row to its left in **A** and the column above it in **B**. In the above figure, square thread blocks of dimension **BLOCK_SIZE × BLOCK_SIZE** and will assume that the dimensions of **A** and **B** are all multiples of **BLOCK_SIZE**. Again, each thread will be responsible for computing one element of the product matrix **C**.

It decomposes matrices **A** and **B** into non-overlapping submatrices of size **BLOCK_SIZE × BLOCK_SIZE**. It shows in above figure in red row and red column. It passes through the same number of these submatrices, since they are of equal length. If it load the left- most of those submatrices of matrix **A** into shared memory, and the top-most of those submatrices of matrix **B** into shared memory, then it compute the first **BLOCK_SIZE** products and add them together just by reading the shared memory. But here is the benefit, as long as it have those submatrices in shared memory, every thread's thread block (computing the BLOCK_SIZE × BLOCK_SIZE submatrix of C) can compute that portion of their sum as well from the same data in shared memory. When each thread has computed this sum, it loads the next **BLOCK_SIZE** × **BLOCK_SIZE** submatrices from **A** and **B**, and continue adding the term-by-term products to our value in **C**.

Applications: 1. Fast Video Trans-coding & Enhancement.

2. Medical Imaging.
3. Neural Networks.
4. Gate-level VLSI Simulation.

INPUT: %%cu

```
#include<iostream>

#include<cstdlib>


using namespace std;


//VectorAdd parallel function

__global__void vectorAdd(int *a, int *b, int *result, int n)

{

    int tid=threadIdx.x+blockIdx.x*blockDim.x;

    if(tid<n)

    {

        result[tid]=a[tid]+b[tid];

    }

}

int main()

{
```

```
int *a,*b,*c;
int *a_dev,*b_dev,*c_dev;
int n=1<<24;

a=new int[n];
b=new  int[n];
c=new int[n];
int *d=new int[n];
int size=n*sizeof(int);
cudaMalloc(&a_dev,size);
cudaMalloc(&b_dev,size);
cudaMalloc(&c_dev,size);

//Array initialization..You can use Randon function to assign values
for(int i=0;i<n;i++)
{
    a[i]=1;
    b[i]=2;
    d[i]=a[i]+b[i]; //calculating serial addition
}


cudaEvent_t  start,end;

cudaEventCreate(&start);
cudaEventCreate(&end);
```

```
}

cudaMemcpy(a_dev,a,size,cudaMemcpyHostToDevice);
cudaMemcpy(b_dev,b,size,cudaMemcpyHostToDevice);
int threads=1024;
int blocks=(n+threads-1)/threads;
cudaEventRecord(start);

//Parallel addition program
vectorAdd<<<blocks,threads>>>(a_dev,b_dev,c_dev,n);

cudaEventRecord(end);
cudaEventSynchronize(end);

float time=0.0;
cudaEventElapsedTime(&time,start,end);

cudaMemcpy(c,c_dev,size,cudaMemcpyDeviceToHost);

//Calculate the error term.
int error=0;
for(int  i=0;i<n;i++){
    error+=d[i]-c[i];
    //cout<<" gpu "<<c[i]<<" CPU "<<d[i];
}

cout<<"Error : "<<error;
```

```
    cout<<"\nTime Elapsed:  "<<time;

    return 0;
}

%%cu

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

//Matrix multiplication Cuda
__global__void matrixMultiplication(int *a, int *b, int *c, int n)
{
    int row=threadIdx.y+blockDim.y*blockIdx.y;
    int col=threadIdx.x+blockDim.x*blockIdx.x;
    int sum=0;

    if(row<n && col<n)
    for(int  j=0;j<n;j++)
    {
        sum=sum+a[row*n+j]*b[j*n+col];
    }
    c[n*row+col]=sum;
}
int main()
{
    int *a,*b,*c;
    int *a_dev,*b_dev,*c_dev;
    int n=3;

    a=new  int[n*n];
    b=new  int[n*n];
    c=new  int[n*n];
    int *d=new int[n*n];
    int size=n*n*sizeof(int);
    cudaMalloc(&a_dev,size);
    cudaMalloc(&b_dev,size);
    cudaMalloc(&c_dev,size);

    //Array initialization
    for(int  i=0;i<n*n;i++)
    {
          a[i]=2; //rand()%n;
        b[i]=1;//rand()%n;
       //  d[i]=a[i]+b[i];
    }
    cudaEvent_t start,end;

    cudaEventCreate(&start);
    cudaEventCreate(&end);

    cudaMemcpy(a_dev,a,size,cudaMemcpyHostToDevice);
    cudaMemcpy(b_dev,b,size,cudaMemcpyHostToDevice);



    dim3 threadsPerBlock(n, n);
    dim3 blocksPerGrid(1, 1);

    if(n*n>512){
        threadsPerBlock.x=512;
        threadsPerBlock.y=512;
        blocksPerGrid.x=ceil((double)n/(double)threadsPerBlock.x);
        blocksPerGrid.y=ceil((double)n/(double)threadsPerBlock.y);
    }
    //GPU Multiplication
    cudaEventRecord(start);

matrixMultiplication<<<blocksPerGrid,threadsPerBlock>>>(a_dev,b_
dev,c_dev,n);

    cudaEventRecord(end);
    cudaEventSynchronize(end);
```

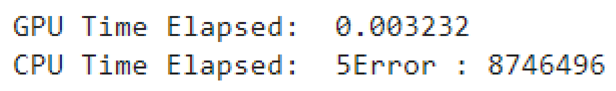
```
float time=0.0;
cudaEventElapsedTime(&time,start,end);

cudaMemcpy(c,c_dev,size,cudaMemcpyDeviceToHost);


//CPU matrix multiplication
int sum=0;
for(int row=0;row<n;row++)
{
    for(int col=0;col<n;col++)
    {
       sum=0;
       for(int k=0;k<n;k++)
        sum=sum+a[row*n+k]*b[k*n+col];
        d[row*n+col]=sum;
    }

}
int error=0;
for(int i=0;i<n*n;i++){
    error+=d[i]-c[i];
    //cout<<" gpu "<<c[i]<<" CPU "<<d[i]<<endl;
}

cout<<"Error : "<<error;
```

```
    cout<<"\nTime Elapsed:  "<<time;


    return 0;

}
```

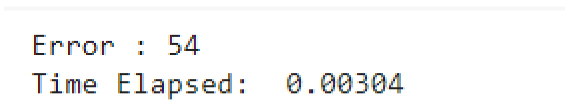
```
%%cu
#include<iostream>
#include<cstdlib>
#include<cmath>
#include<time.h>
using namespace std;


__global__void matrixVectorMultiplication(int *a, int *b,
int *c, int n)
{
    int row=threadIdx.x+blockDim.x*blockIdx.x;
    int sum=0;

    if(row<n)
    for(int j=0;j<n;j++)
    {
        sum=sum+a[row*n+j]*b[j];
    }

    c[row]=sum;
}
int main()
{
    int *a,*b,*c;
    int *a_dev,*b_dev,*c_dev;
    int n=32;

    a=new int[n*n];
    b=new int[n];
    c=new int[n];
    int *d=new int[n];
    int size=n*sizeof(int);
    cudaMalloc(&a_dev,size*size);
    cudaMalloc(&b_dev,size);
    cudaMalloc(&c_dev,size);

    for(int i=0;i<n;i++)
    {
        for(int j=0;j<n;j++)
        {
            a[i*n+j]= i*n+j+1; //rand()%n;

        }

        b[i]= i+1; //rand()%n;
        //cout<<a[i]<<" ";
       // d[i]=a[i]+b[i];
    }


    cudaEvent_t start,end;

    cudaEventCreate(&start);
    cudaEventCreate(&end);

    cudaMemcpy(a_dev,a,size*size,cudaMemcpyHostToDevice);
    cudaMemcpy(b_dev,b,size,cudaMemcpyHostToDevice);


    dim3 threadsPerBlock(n, n);
    dim3 blocksPerGrid(1, 1);

    if(n*n>512){
        threadsPerBlock.x=512;
        threadsPerBlock.y=512;
        blocksPerGrid.x=ceil((double)n/
(double)threadsPerBlock.x);
        blocksPerGrid.y=ceil((double)n/
(double)threadsPerBlock.y);
    }

    cudaEventRecord(start);
    matrixVectorMultiplication<<<blocksPerGrid,threadsPerBl
ock>>>(a_dev,b_dev,c_dev,n);

    cudaEventRecord(end);
    cudaEventSynchronize(end);

    float time=0.0;
    cudaEventElapsedTime(&time,start,end);

    cudaMemcpy(c,c_dev,size,cudaMemcpyDeviceToHost);
    cout<<"\nGPU Time Elapsed:  "<<time;

    //CPU matrixVector multiplication
    clock_t t=clock();
    int sum=0;
    for(int row=0;row<n;row++)
    {
        sum=0;
        for(int col=0;col<n;col++)
        {
              sum=sum+a[row*n+col]*b[col];

        }
      d[row]=sum;
    }
    t=clock()-t;
        cout<<"\nCPU Time Elapsed:  "<<((double)t);      //
((double)t)/CLOCKS_PER_SEC;


    int error=0;
    for(int i=0;i<n;i++){
        error+=d[i]-c[i];
       // cout<<" gpu "<<c[i]<<" CPU "<<d[i]<<endl;
    }

    cout<<"Error : "<<error;


    return 0;
}
```

Output:

Matrix vector Multiplication

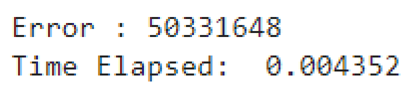


```
GPU Time Elapsed:  0.003232
CPU Time Elapsed:  5Error : 8746496
```



```
Error : 54
Time Elapsed:  0.00304
```

Matrix Multiplication



```
Error : 50331648
Time Elapsed:  0.004352
```

Vector Addition

Conclusion: Strassen’s Matrix Multiplication Algorithm have been implemented parallel using GPU computing in the CUDA environment

Name: Vikas Mane

Class: BE-B

Roll Number: B1921152

PRN Number: 72000291F

Assignment No: 7

Title: For Bubble Sort and Merger Sort, based on existing sequential algorithms, design and implement parallel algorithm utilizing all resources available.

Aim: Understand Parallel Sorting Algorithms like Bubble sort and Merge Sort.

Prerequisites: Student should know basic concepts of Bubble sort and Merge Sort.

Objective: Study of Parallel Sorting Algorithms like Bubble sort and Merge Sort

Theory:

i) What is Sorting? • Arrange elements of a list into certain order

• Make data become easier to access

• Speed up other operations such as searching and merging. Many sorting algorithms with different time and space complexities

• Based on an existing sequential sort algorithm – Try to utilize all resources available

– Possible to turn a poor sequential algorithm into a reasonable parallel algorithm (bubble sort and parallel bubble sort)

• Completely new approach – New algorithm from scratch

– Harder to develop

– Sometimes yield better solution

Sorting is a process of arranging elements in a group in a particular order, i.e., ascending order, descending order, alphabetic order, etc.

ii) What is Parallel Sorting?
A sequential sorting algorithm may not be efficient enough when we have to sort a huge volume of data. Therefore, parallel algorithms are used in sorting.

• Based on an existing sequential sort algorithm – Try to utilize all resources available

– Possible to turn a poor sequential algorithm into a reasonable parallel algorithm (bubble sort and parallel bubble sort)

• Completely new approach – New algorithm from scratch

– Harder to develop

– Sometimes yield better solution

Bubble Sort
The idea of bubble sort is to compare two adjacent elements. If they are not in the right order,switch them. Do this comparing and switching (if necessary) until the end of the array is reached. Repeat this process from the beginning of the array n times.

• One of the straight-forward sorting methods – Cycles through the list

– Compares consecutive elements and swaps them if necessary

– Stops when no more out of order pair
• Slow & inefficient
• Average performance is O(n2)

Bubble Sort Example
Here we want to sort an array containing [8, 5, 1]. The following figure shows how we can sortthis array using bubble sort. The elements in consideration are shown in **bold.**

8, 5, 1	Switch 8 and 5
5, **8, 1**	Switch 8 and 1
5, 1, 8	Reached end start again.
5, 1, 8	Switch 5 and 1
1, **5, 8**	No Switch for 5 and 8
1, 5, 8	Reached end start again.
1, 5, 8	No switch for 1, 5
1, **5, 8**	No switch for 5, 8
1, 5, 8	Reached end.

• • Implemented as a pipeline.
• • Let local_size = n / no_proc. We divide the array in no_proc parts, and each process executes the bubble sort on its part, including comparing the last element with the first one belonging to the next thread.
• • Implement with the loop (instead of j<i) for (j=0; j<n-1; j++)
• • For every iteration of i, each thread needs to wait until the previous thread has finished that iteration before starting.
• • We'll coordinate using a barrier.

Algorithm for Parallel Bubble Sort

1. For *k* = 0 to *n*-2
2. If *k* is even then

```
3. for $i$ = 0 to ($n$/2)-1 do in parallel
4. If $A[2i] > A[2i+1]$ then
5. Exchange $A[2i] \leftrightarrow A[2i+1]$
6. Else
7. for $i$ = 0 to ($n$/2)-2 do in parallel 8. If $A[2i+1] > A[2i+2]$ then
9. Exchange $A[2i+1] \leftrightarrow A[2i+2]$
10. Next $k$
```

Merge Sort

• Collects sorted list onto one processor

• Merges elements as they come together

• Simple tree structure

• Parallelism is limited when near the root

Theory:

To sort A[p .. r]:

1. Divide Step

If a given array *A* has zero or one element, simply return; it is already sorted. Otherwise, split*A*[*p* .. *r*] into two subarrays*A*[*p* .. *q*] and *A*[*q* + 1 .. *r*], each containing about half of the elements of *A*[*p* .. *r*]. That is, *q* is the halfway point of *A*[*p* .. *r*].

2. Conquer Step

Conquer by recursively sorting the two subarrays*A*[*p* .. *q*] and *A*[*q* + 1 .. *r*].

3. Combine Step

Combine the elements back in *A*[*p* .. *r*] by merging the two sorted subarrays*A*[*p* .. *q*] and *A*[*q* + 1 .. *r*] into a sorted sequence. To accomplish this step, we will define a procedure MERGE (*A*, *p*, *q*, *r*).

Parallel Merge Sort

• Parallelize processing of sub-problems

• Max parallelization achived with one processor per node (at each layer/height)

Algorithm for Parallel Merge Sort

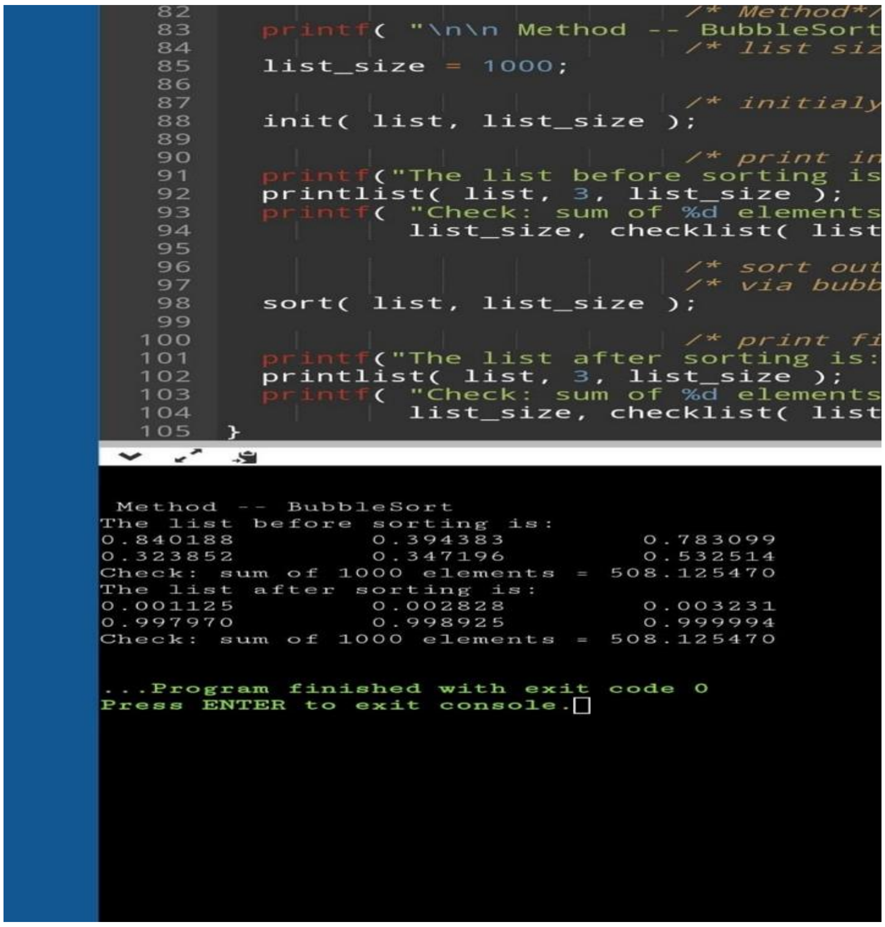
```
1. Procedure parallelMergeSort
2. Begin
3. Create processors Pi where i = 1 to n
4. if i > 0 then recieve size and parent from the root
5. recieve the list, size and parent from the root
6. endif
7. midvalue= listsize/2
8. if both children is present in the tree then
9. send midvalue, first child
10. send listsize-mid,second child
11. send list, midvalue, first child
12. send list from midvalue, listsize-midvalue, second child
13. call mergelist(list,0,midvalue,list, midvalue+1,listsize,temp,0,listsize)
```

```
14. store temp in another array list2
15. else
16. call parallelMergeSort(list,0,listsize)
17. endif
18. if i >0 then
19. send list, listsize,parent
20. endif
21. end
```

INPUT:

1. Array of integer numbers.

OUTPUT:



```
                                            /* Method*/
    printf( "\n\n Method -- BubbleSort
                                            /* list siz
    list_size = 1000;

                                            /* initialy
    init( list, list_size );

                                            /* print in
    printf("The list before sorting is
    printlist( list, 3, list_size );
    printf( "Check: sum of %d elements
              list_size, checklist( list

                                            /* sort out
                                            /* via bubb
    sort( list, list_size );

                                            /* print fi
    printf("The list after sorting is:
    printlist( list, 3, list_size );
    printf( "Check: sum of %d elements
              list_size, checklist( list
}
```

```
 Method -- BubbleSort
The list before sorting is:
0.840188           0.394383           0.783099
0.323852           0.347196           0.532514
Check: sum of 1000 elements = 508.125470
The list after sorting is:
0.001125           0.002828           0.003231
0.997970           0.998925           0.999994
Check: sum of 1000 elements = 508.125470


...Program finished with exit code 0
Press ENTER to exit console.
```

Name: Vikas Mane

Class: BE-B

Roll Number: B1921152

PRN Number: 72000291F

Assignment 8

Title: Implement Tic-Tac-Toe using A* Algorithm

System Requirements: NetBeans IDE

Theory:

· A* is a computer algorithm that is widely used in pathfinding and graph traversal, the process of plotting an efficiently traversable path between multiple points, called nodes. It is complete and will always find a solution if one exists. It evaluates nodes by combining g(n), the cost to reach the node, and h(n), the cost to get from the node to the goal:

f (n) = g(n) + h(n)

· Since g(n) gives the path cost from the start node to node n, and h(n) is the estimated cost of the cheapest path from n to the goal, we have f (n) = estimated cost of the cheapest solution through n. Thus, if we are trying to find the cheapest solution, a reasonable thing to try first is the node with the lowest value of g(n) + h(n). The algorithm is identical to UNIFORM-COST-SEARCH except that A* uses g + h instead of g

· A* uses a best-first search and finds a least-cost path from a given initial node to one goal node (out of one or more possible goals). As A* traverses the graph, it follows a path of the lowest known heuristic cost, keeping a sorted priority queue of alternate path segments along the way

· It uses a distance-plus-cost heuristic function (usually denoted) to determine the order in which the search visits nodes in the tree. The distance-plus-cost heuristic is a sum of two functions: the path-cost function, which is the cost from the starting node to the current node (usually denoted) an admissible "heuristic estimate" of the distance to the goal (usually denoted ***h(x)***)

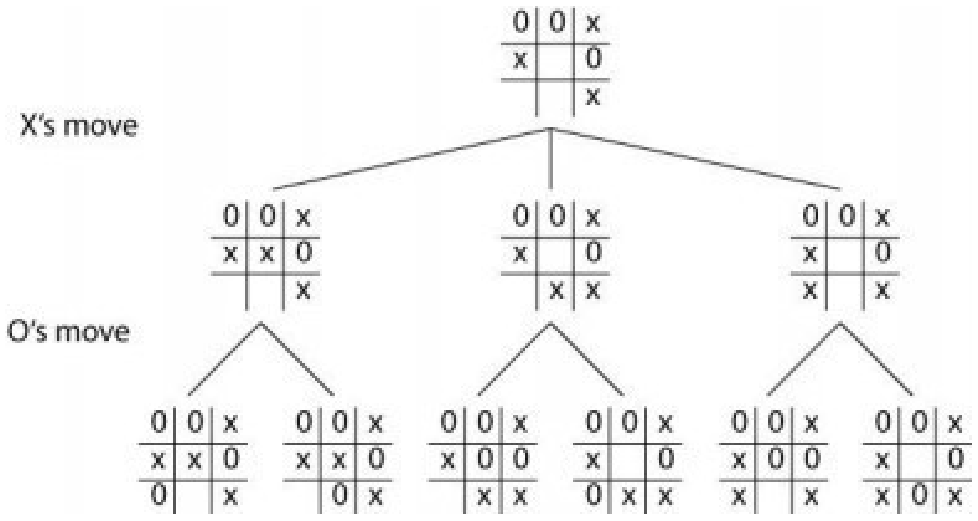
· The part of the function must be an admissible heuristic; that is, it must not overestimate the distance to the goal. Thus, for an application like routing, might represent the straight-line distance to the goal, since that is physically the smallest possible distance between any two points or nodes.If the heuristic h satisfies the additional condition ***h(x) <= d(x,y) + h(y)*** for every edge x, y of the graph (where d denotes the length of that edge), then h is called monotone, or consistent. In such a case, A* can be implemented more efficiently—roughly speaking, no node needs to be processed more than once (see closed set below) and A* is equivalent to running Dijkstra's algorithm with the reduced cost.

d'(x,y) := d(x,y) - h(x) + h(y)

· For example, since airline distance never overestimates actual highway distance, and manhattan distance never overestimates actual moves in the gliding tile.

Tic-Tac-Toe:

Tic-tac-toe (also known as noughts and crosses or Xs and Os) is a paper-and-pencil game for two players, X and O, who take turns marking the spaces in a 3×3 grid. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row wins the game.



f(n)=g(n)+h(n)

Where, f(n) is evaluation function g(n) is path cost h(n) is heuristic function

A* is commonly used for the common path finding problem in applications such as games, but was originally designed as a general graph traversal algorithm.

Objective: Use A* algorithm to implement Tic Tac Toe problem

Input: Enter X or 0 then enter position you want to place X or 0 (1 to 9)

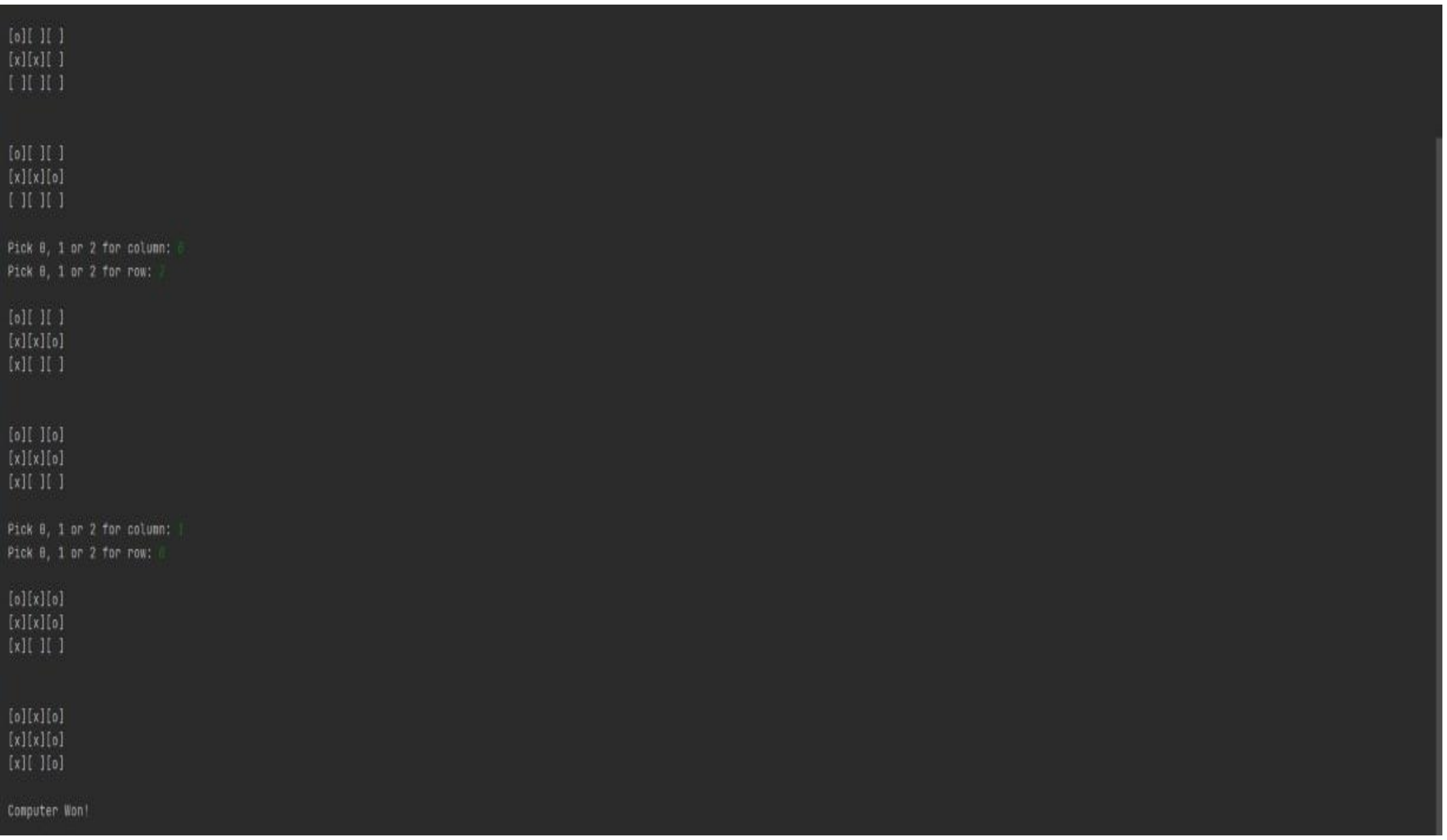
Output: Draw, you win or AI Player wins!

Result / Conclusion: Using A* algorithm we are able to solve Tic Tac Toe problem.

Program:

```
public static void main(String args[]) {
        init();
        selectSymbol();
        play();
    }
public static void play() {                              //keep playing while
value returned from checkWin() is "nothing"
        moveCount = 0;
        printBoard();
        while (checkWin(board) == "nothing") {
            if (isHumanTurn()) {
                inputMove();
            } else {
                aiMove();
            }
            printBoard();
            moveCount++;
        }
        if (checkWin(board) == humanPlayer) {
            System.out.println("You win!");
        } else if (checkWin(board) == aiPlayer) {
            System.out.println("AI Player wins!");
        } else {
            System.out.println("Draw!");
        }
    }
```

Output:



```
[o][ ][ ]
[x][x][ ]
[ ][ ][ ]


[o][ ][ ]
[x][x][o]
[ ][ ][ ]

Pick 0, 1 or 2 for column: 0
Pick 0, 1 or 2 for row: 2

[o][ ][ ]
[x][x][o]
[x][ ][ ]


[o][ ][o]
[x][x][o]
[x][ ][ ]

Pick 0, 1 or 2 for column: 1
Pick 0, 1 or 2 for row: 0

[o][x][o]
[x][x][o]
[x][ ][ ]


[o][x][o]
[x][x][o]
[x][ ][o]

Computer Won!
```

Name: Vikas Mane

Class: BE-B

Roll Number: B1921152

PRN Number: 72000291F

Assignment 9

Title: Solve 8-puzzle problem using A* algorithm. Assume any initial configuration and define goal configuration clearly.
System Requirements: NetBeans IDE

Theory:

- In computer science, A* (pronounced as "A star") is a computer algorithm that is widely used in path finding and graph traversal, the process of plotting an efficiently traversable path between multiple points, called nodes. The A* algorithm combines features of uniform-cost search and pure heuristic search to efficiently compute optimal solutions.
- A* algorithm is a best-first search algorithm in which the cost associated with a node is f(n) = g(n) + h(n), where g(n) is the cost of the path from the initial state to node n and h(n) is the heuristic estimate or the cost or a path from node n to a goal.
- Thus, f(n) estimates the lowest total cost of any solution path going through node n. At each point a node with lowest f value is chosen for expansion. Ties among nodes of equal f value should be broken in favor of nodes with lower h values. The algorithm terminates when a goal is chosen for expansion.
  - A* algorithm guides an optimal path to a goal if the heuristic function h(n) is admissible, meaning it never overestimates actual cost. For example, since airline distance never overestimates actual highway distance, and Manhattan distance never overestimates actual moves in the gliding tile.
- The main drawback of A* algorithm and indeed of any best-first search is its memory requirement. Since at least the entire open list must be saved, A* algorithm is severely spacelimited in practice, and is no more practical than best-first search algorithm on current machines. For example, while it can be run successfully on the eight puzzles, it exhausts available memory in a matter of minutes on the fifteen puzzles.
- A star algorithm is very good search method, but with complexity problems
- To implement such a graph-search procedure, we will need to use two lists of node:
  1) **_OPEN:** nodes that have been generated and have had the heuristic function applied to them but which have not yet been examined (i.e., had their successors generated). OPEN is actually a priority queue in which the elements with the highest priority are those with the most promising value of the heuristic function.
  2) **_CLOSED:** Nodes that have already been examined. We need to keep these nodes in memory if we want to search a graph rather than a tree, since whether a node is generated; we need to check whether it has been generated before.

A * Algorithm:

1. Put the start nodes on OPEN.
2. If OPEN is empty, exit with failure

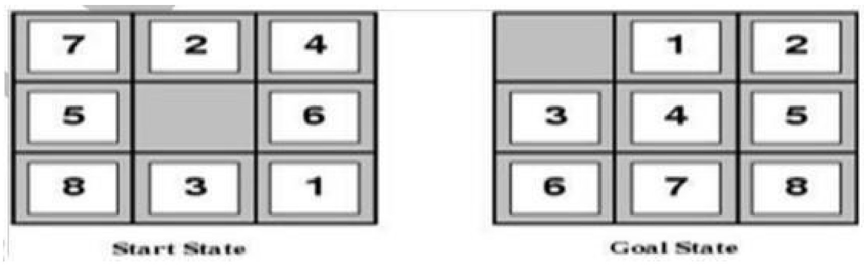
3. Remove from OPEN and place on CLOSED a node n having minimum f
4. If n is goal node exit successfully with a solution path obtained by tracing back the ptrs from n to s

5. Otherwise, expand n generating its children and directing pointers from each child node to n
   - For every child node n’ do
   - Evaluate h(n’) and compute f(n’) = g(n’) +h(n’)= g(n)+c(n, n’)+h(n)
   - If n’ is already on OPEN or CLOSED compare its new f with the old f and attach the lowest f to n’ put n’ with its f value in the right order in OPEN

6. Go to step 2.

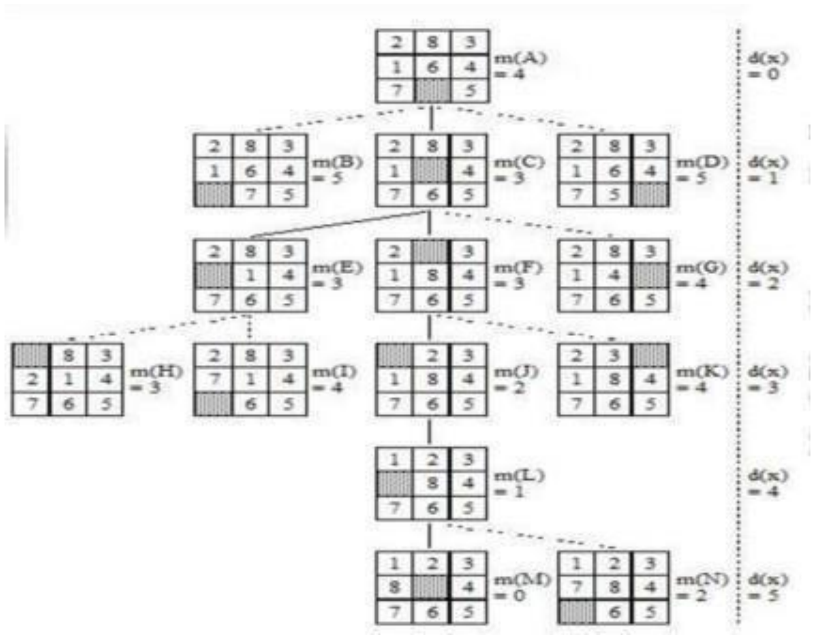
Example of calculation of heuristic values for 8-puzzle problem:

h1(n) = number of misplaced tiles
h2(n) =no. of squares from desired location of each tile



h1(S) = 8
h2(S) = 3+1+2+2+2+3+3+2 = 18

Implementation logic for 8-puzzle problem using A* algorithm



f(n)=g(n)+h(n)

Where, f(n) is evaluation function g(n) is path cost h(n) is heuristic function

A* is commonly used for the common path finding problem in applications such as games, but was originally designed as a general graph traversal algorithm.

Objective:

Student will learn:

i) The Basic Concepts of a Star: Evaluation function, Path Cost, Heuristic function, Calculation of heuristic function
ii) General structure of eight puzzle problem
iii) Logic of A star implementation for eight puzzle problems

Outcome: 8 puzzle problem is successfully solved using A* algorithm.

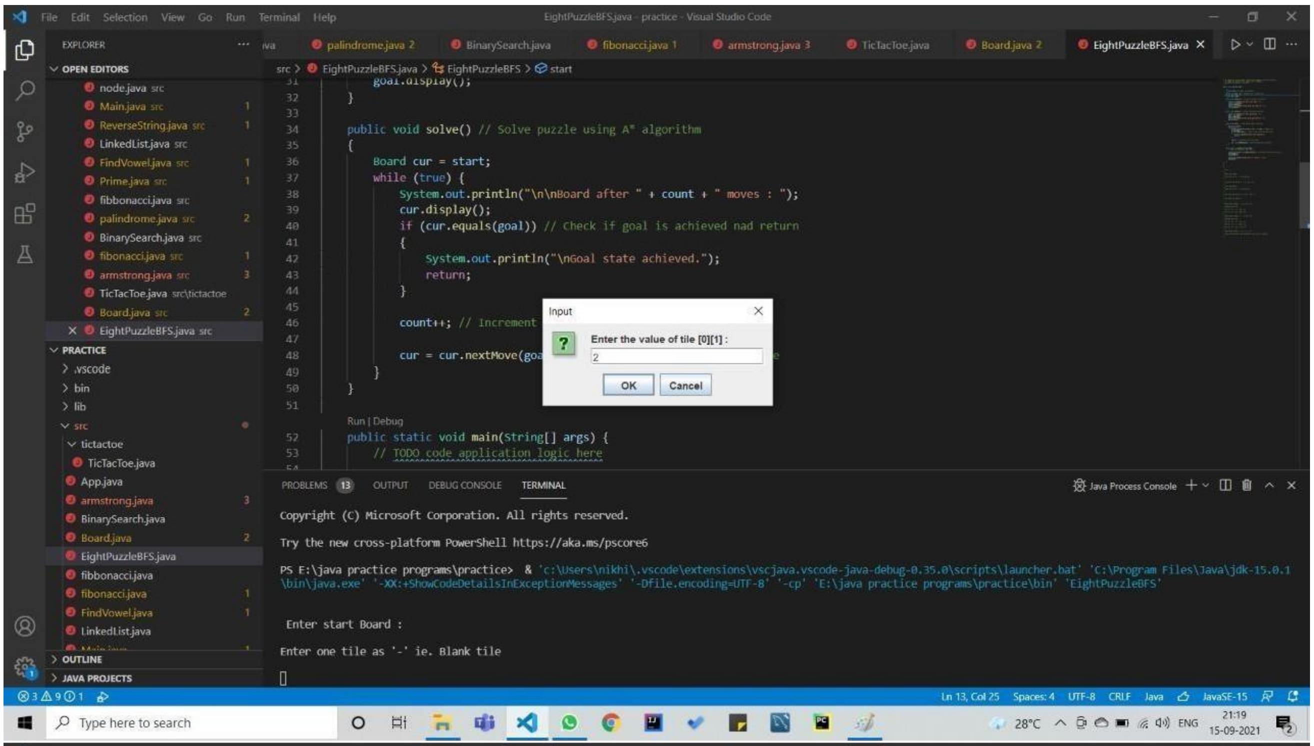
Input: Enter the first 9 elements of matrix

Output: Board after 0 moves matrix will get displayed

Result / Conclusion: A star algorithm is implemented for eight puzzle problems

Program:
```
package eughtpuzzle; public
class EightPuzzleBFS {
    private int count=0;                                                    //
Initialize no. of moves to 0
    private Board start;
    private Board goal; public
    void initStart()
//Accept and display start board
    {
        System.out.println("\n\n Enter start Board : ");
        start=new Board(); start.initBoard();
        System.out.println("\n\nThe given start board is : ");
        start.display();
    }
    public void initGoal()
//Accept and display goal board
    {
        System.out.println("\n\n Enter goal Board : ");
        goal=new Board(); goal.initBoard();
        System.out.println("\n\nThe given goal board is : ");
        goal.display();
    }

    public void solve()                                              // Solve
puzzle using A* algorithm
    {
        Board cur = start; while(true)
        {
            System.out.println("\n\nBoard after "+count+" moves : ");
            cur.display();
            if(cur.equals(goal))                                    //Check if
goal is achieved nad return
            {
                System.out.println("\nGoal state achieved.");
            return; }
            count++;                                                 //
Increment count as per moves
            cur = cur.nextMove(goal);                          // get the board
after next move
        }
    }
public static void main(String[] args) {

        EightPuzzleBFS ep = new EightPuzzleBFS();                            //
Instantiate and solve the puzzle
        ep.initStart(); ep.initGoal();
        System.out.println("\n\nThe board is solved as : \n"); ep.solve();
    }
}
```

Output:



```
    goal.display();
    }

    public void solve() // Solve puzzle using A* algorithm
    {
        Board cur = start;
        while (true) {
            System.out.println("\n\nBoard after " + count + " moves : ");
            cur.display();
            if (cur.equals(goal)) // Check if goal is achieved nad return
            {
                System.out.println("\nGoal state achieved.");
                return;
            }

            count++; // Increment

            cur = cur.nextMove(goa
        }
    }

    public static void main(String[] args) {
        // TODO code application logic here
```

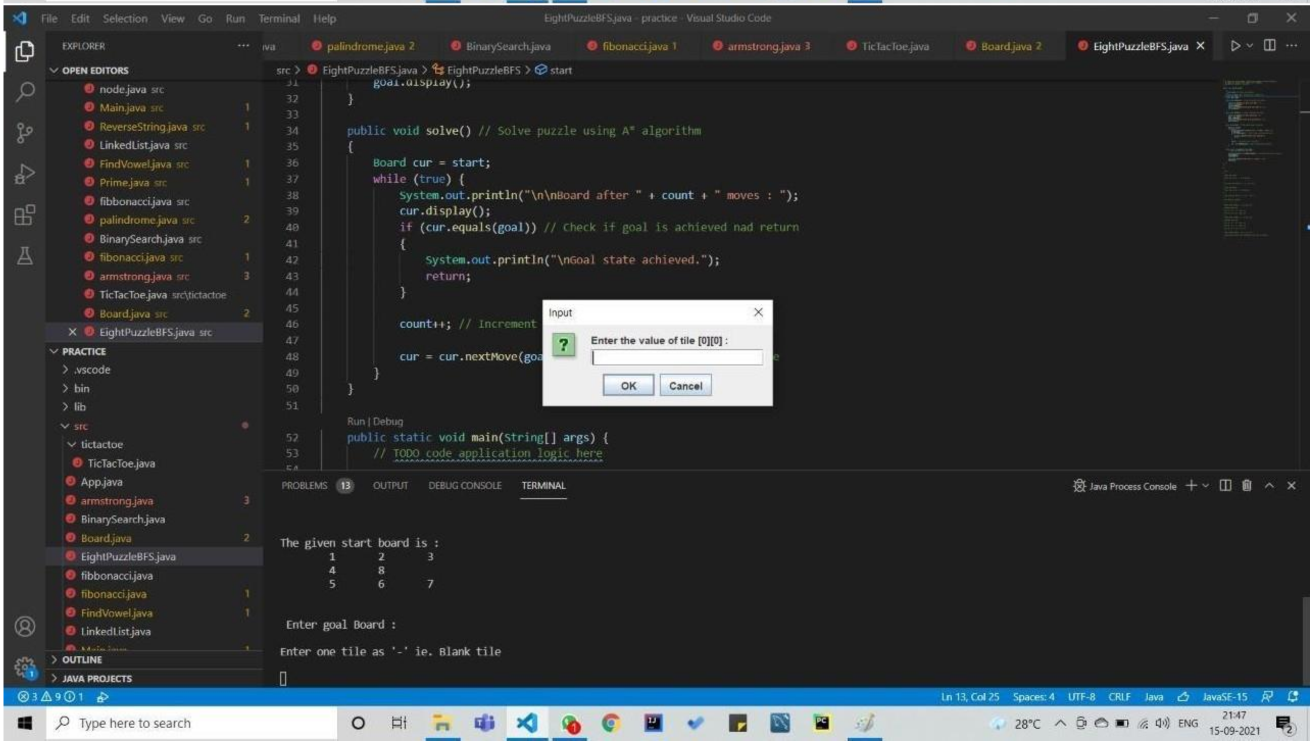
Input

Enter the value of tile [0][1] :

2

OK Cancel

```
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS E:\java practice programs\practice>  & 'c:\Users\nikhi\.vscode\extensions\vscjava.vscode-java-debug-0.35.0\scripts\launcher.bat' 'C:\Program Files\Java\jdk-15.0.1
\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-Dfile.encoding=UTF-8' '-cp' 'E:\java practice programs\practice\bin' 'EightPuzzleBFS'


 Enter start Board :

Enter one tile as '-' ie. Blank tile
```



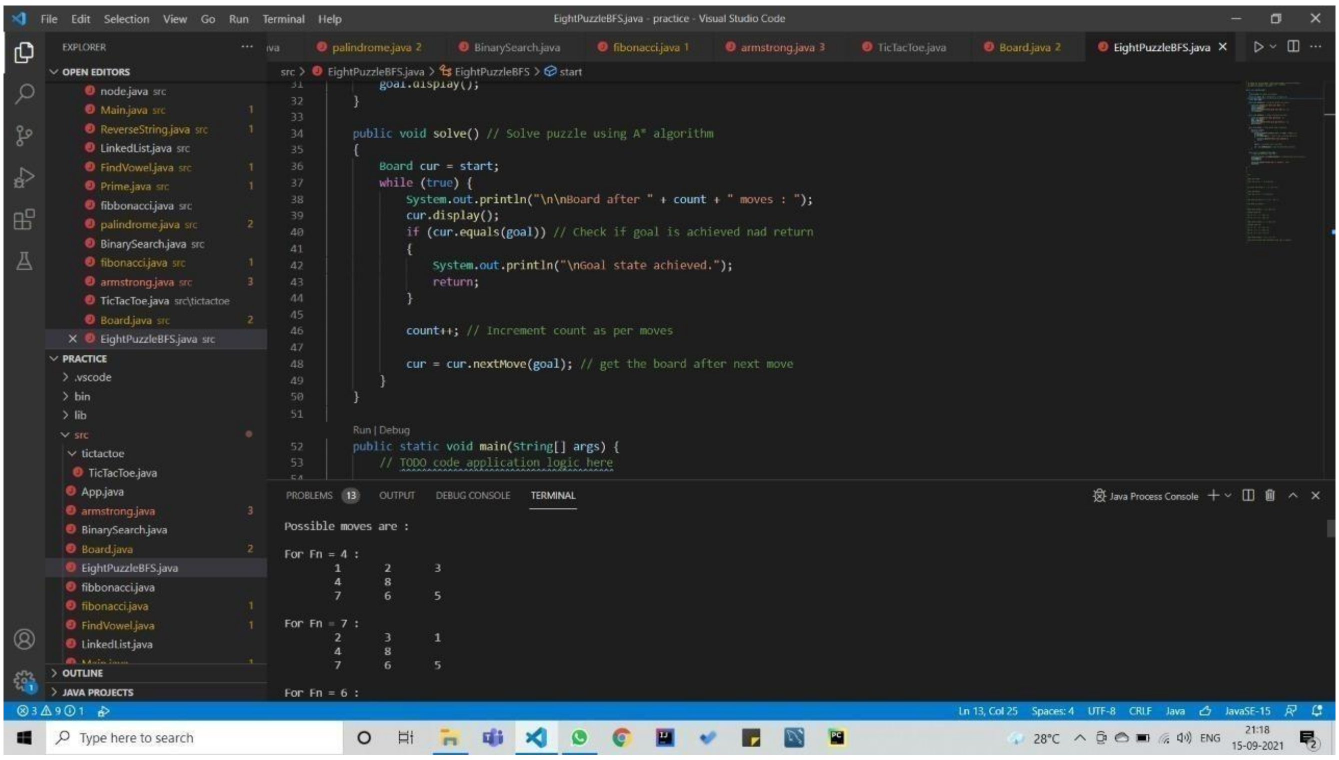
```
    goal.display();
    }

    public void solve() // Solve puzzle using A* algorithm
    {
        Board cur = start;
        while (true) {
            System.out.println("\n\nBoard after " + count + " moves : ");
            cur.display();
            if (cur.equals(goal)) // Check if goal is achieved nad return
            {
                System.out.println("\nGoal state achieved.");
                return;
            }

            count++; // Increment

            cur = cur.nextMove(goa
        }
    }

    public static void main(String[] args) {
        // TODO code application logic here
```

Input

Enter the value of tile [0][0] :

OK Cancel

```
The given start board is :
        1       2       3
        4       8
        5       6       7


 Enter goal Board :

Enter one tile as '-' ie. Blank tile
```



```java
        goal.display();
    }

    public void solve() // Solve puzzle using A* algorithm
    {
        Board cur = start;
        while (true) {
            System.out.println("\n\nBoard after " + count + " moves : ");
            cur.display();
            if (cur.equals(goal)) // Check if goal is achieved nad return
            {
                System.out.println("\nGoal state achieved.");
                return;
            }

            count++; // Increment count as per moves

            cur = cur.nextMove(goal); // get the board after next move
        }
    }

    Run | Debug
    public static void main(String[] args) {
        // TODO code application logic here
```

```
Possible moves are :

For Fn = 4 :
        1       2       3
        4       8
        7       6       5

For Fn = 7 :
        2       3       1
        4       8
        7       6       5

For Fn = 6 :
```

Name: Vikas Mane
Roll No: B1921152
PRN : 72000291F
Class: BE-B

Assignment 10

Title: Develop elementary chatbot for suggesting investment as per the customer needs.

System Requirements: Java JDK, Visual Studio/Netbeans

Theory:

- A chatbot (also known as a talkbot, chatterbot, Bot, IM bot, interactive agent, or Artificial Conversational Entity) is a computer program or an artificial intelligence which conducts a conversation via auditory or textual methods. Such programs are often designed to convincingly simulate how a human would behave as a conversational partner, thereby passing the Turing test. Chatbots are typically used in dialog systems for various practical purposes including customer service or information acquisition. Some chatterbots use sophisticated natural language processing systems, but many simpler systems scan for keywords within the input, then pull a reply with the most matching keywords, or the most similar wording pattern, from a database.
- The term "ChatterBot" was originally coined by Michael Mauldin (creator of the first Verbot, Julia) in 1994 to describe these conversational programs. Today, most chatbots are either accessed via virtual assistants such as Google Assistant and Amazon Alexa, via messaging apps such as Facebook Messenger or WeChat, or via individual organizations' apps and websites. Chatbots can be classified into usage categories such as conversational commerce (e-commerce via chat), analytics, communication, customer support, design, developer tools, education, entertainment, finance, food, games, health, HR, marketing, news, personal, productivity, shopping, social, sports, travel and utilities.

Chatbots Work on Pattern Recognition and a Set of Algorithms:

Most chatbots work on a basic model of:

1. **Entities:** The thing the user is talking about.
2. **Intents:** The question the user asks the chatbot.
3. **Responses:** The answer the chatbot provides. It identifies the appropriate predefined responses from its repository. It then selects the most appropriate answer based on the intent and context. The repository is often confined to linguistic building blocks, but the real value comes from structured information it can mine from elsewhere.

Creating a Chatbot that Uses AI

- These systems use machine learning and natural language processing (NLP) to interpret text like a person and find the right answer. They can also use visual information, such as facial recognition and images. Companies such as Amazon, Google, IBM and Microsoft, offer machine learning AI platforms you can use to power your own AI-driven chatbot.

Natural Language Processing

- Natural language processing involves breaking down sentences and other parts of language, into components. It processes the semantics of the content to identify things like the entities and intents of the user. This has been a hard problem for computer science to tackle, but recent advances in the field have made this feasible.

Machine Learning

- These systems use a corpus (a body of content) to retrieve the correct answer. Machine learning works best with massive repositories. So it will often include open-source repositories, such as structured Wikipedia. It uses algorithms to manipulate the data - particularly, linguistic pattern matching to find and score the information in the corpus. It continues to improve its responses based on its interactions with users. In other words, the machine learns.

Feeding the Bots

- To help the machine learn, subject matter experts train it by uploading pairs of questions and answers. This enables the system to check the results of its learning for accuracy against the real world. This is called the ground truth. It can apply this learning to any new information added to the corpus or through user feedback.
- You can also use unstructured content to help the system learn, where it contains relevant information. This might be in articles and blog posts. Platforms such as IBM Watson can process a corpus of unstructured content. However, you do need structured content when you're developing the system - as those question and answer key pairs.

Objective: To deveop an elementary chatbot using Java

Outcome: An elementary chatbot is designed.

Input: Keywords such as “hii”, “hello”, “banks”, “finance”, “shares”, etc.

Output: Output by chatbots based on different inputs by user.

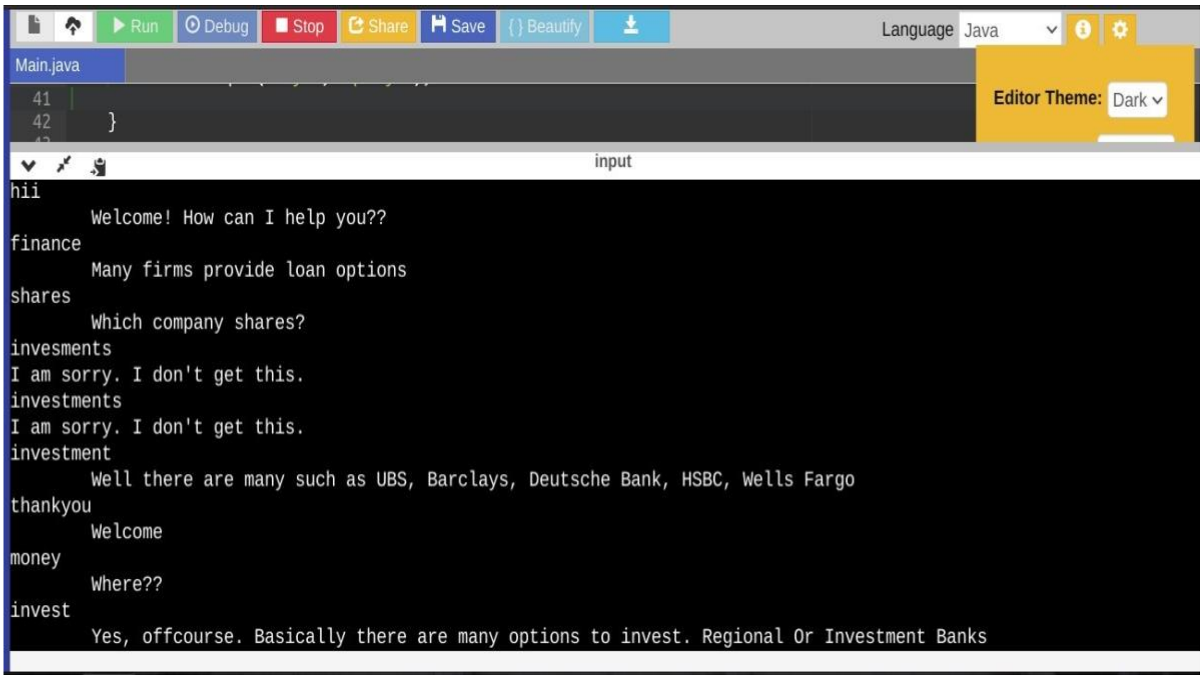
Result / Conclusion: Chatbot is the functionality used for conversation in a textual format. A chatbot is designed by using Java.

Program:

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;
import java.util.Random;
import java.util.Scanner;

public class chatbot{
    Map<String, String> nouns = new HashMap<String, String>();
    Map<String, String> welcome = new HashMap<String, String>();
    Map<String, String> farewell = new HashMap<String, String>();

    boolean exit = false;
    ArrayList<String> keywords;
    public chatbot(){
        nouns.put("money", "\tWhere??");
        nouns.put("finance", "\tMany firms provide loan options");
        nouns.put("invest", "\tYes, offcourse. Basically there are many op
tions to invest. Regional Or Investment Banks?");
        nouns.put("shares", "\tWhich company shares?");
        nouns.put("loan", "\tWhich loan? Housing ,Personal,Educational. I
recommend to visit SBI website for this");
        nouns.put("investment", "\tWell there are many such as UBS, Barcla
ys, Deutsche Bank, HSBC, Wells Fargo");
        nouns.put("regional", "\tThere are many SBI, IDBI, Kotak Mahindra")
;

        welcome.put("hii", "\tWelcome! How can I help you??");
        welcome.put("hey", "\tHey, how I can help you??");
        welcome.put("hello", "\tHello,How can I help you??");
        welcome.put("thankyou", "\tWelcome");

        farewell.put("bye", "\tBye");
        farewell.put("bbye", "\tBbye");

    }
```

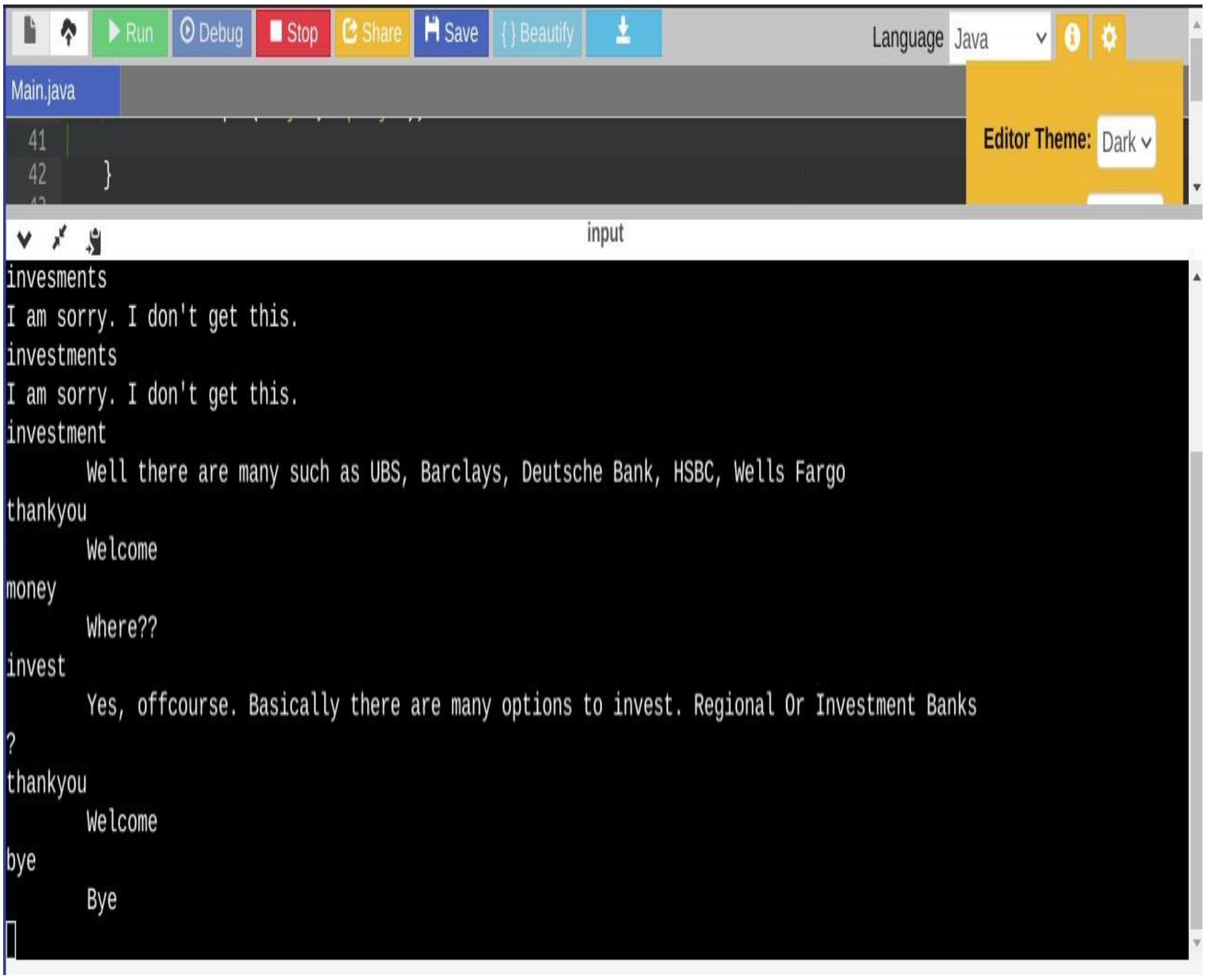
```
        public static void main(String[] args) {

        Scanner s = new Scanner(System.in);
        chatbot c = new chatbot();

        while(true) {
            String input = s.nextLine();

            String output = c.giveans(input);

            System.out.println(output);

            if (c.exit) {
                break;
            }
        }
    }

    public String giveans(String input) {
        Random rand = new Random();
        keywords = new ArrayList<String>();
        String tokens[] = input.split("\\s");

        for (int i = 0; i < tokens.length; i++) {
            if (welcome.containsKey(tokens[i].toLowerCase())) {
                return welcome.get(tokens[i]);

            } else if (farewell.containsKey(tokens[i].toLowerCase())) {
                return farewell.get(tokens[i]);
            } else if (nouns.containsKey(tokens[i].toLowerCase())) {
                return nouns.get(tokens[i]);
            }
        }
        return ("I am sorry. I don't get this.");
    }
}
```

Output:



```
    }
```

```
hii
        Welcome! How can I help you??
finance
        Many firms provide loan options
shares
        Which company shares?
invesments
I am sorry. I don't get this.
investments
I am sorry. I don't get this.
investment
        Well there are many such as UBS, Barclays, Deutsche Bank, HSBC, Wells Fargo
thankyou
        Welcome
money
        Where??
invest
        Yes, offcourse. Basically there are many options to invest. Regional Or Investment Banks
```



```
    }
```

```
invesments
I am sorry. I don't get this.
investments
I am sorry. I don't get this.
investment
        Well there are many such as UBS, Barclays, Deutsche Bank, HSBC, Wells Fargo
thankyou
        Welcome
money
        Where??
invest
        Yes, offcourse. Basically there are many options to invest. Regional Or Investment Banks
?
thankyou
        Welcome
bye
        Bye
```

Name: Vikas ManeRoll
No: B1921152
Class: BE-B

Assignment 11

Title: Implement goal stack planning for the following configurations from the blocks world,

System Requirements: Java JDK, Visual Studio/Netbeans

Theory:

- Planning is process of determining various actions that often lead to a solution.
- Planning is useful for non-decomposable problems where subgoals often interact.
- Goal Stack Planning (in short GSP) is the one of the simplest planning algorithm that is designed to handle problems having compound goals. And it utilizes STRIP as a formal language for specifying and manipulating the world with which it is working.
- This approach uses a Stack for plan generation. The stack can contain Sub-goal and actions described using predicates. The Sub-goals can be solved one by one in any order.

Algorithm:

```
Push the Goal state in to the Stack
Push the individual Predicates of the Goal State into the Stack Loop till the Stack is empty
    Pop an element E from the stack
    IF E is a Predicate
        IF E is True then
                Do Nothing
        ELSE
                Push the relevant action into the Stack
                Push the individual predicates of the Precondition of the action into the Stack
    ELSE IF E is an Action
        Apply the action to the current State.
        Add the action ‘a’ to the plan
```

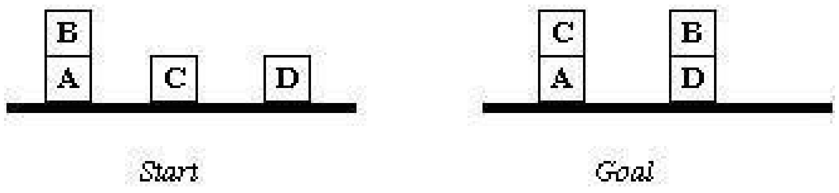
- The Goal Stack Planning Algorithms works will the stack. It starts by pushing the unsatisfied goals into the stack. Then it pushes the individual subgoals into the stack and its pops an element out of the stack. When popping an element out of the stack the element could be either a predicate describing a situation about our world or it could be an action that can be applied to our world under consideration. So based on the kind of element we are popping out from the stack a decision has to be made. If it is a Predicate. Then compares it with the description of the current world, if it is satisfied or is already present in our current situation then there is nothing to do because already its true. On the contrary if the Predicate is not true then we have to select and push relevant action satisfying the predicate to the Stack.
- So after pushing the relevant action into the stack its precondition should also has to be pushed into the stack. In order to apply an operation its precondition has to be satisfied. In other words the present situation of the world should be suitable enough to apply an operation. For that, the preconditions are pushed into the stack once after an action is pushed.

Objective: Student will learn:

I) The Basic Concepts of Goal Stack Planning
II) General Algorithm for Goal Stack Planning
III) Logic of goal stack planning implementation

Input:

Consider the following where wish to proceed from the start to goal state.



No of Blocks : 4

Initial stage : (on b a)^(ontable c)^(ontable a)^(ontable d)^(clear b)^(clear c)^(clear d)^(AE)

Final stage : (on c a)^(on b d)^(ontable a)^(ontable d)^(clear c)^(clear b)^(AE)

Output:

Set of actions to be taken:
1. (unstack b d)
2. (stack b d)
3. (pick c)
4. (stack c a)

Result / Conclusion: Hence we have implemented Goal Stack Planning Algorithm

Program:

```
tab = []
result = []
goalList = ["a", "b", "c", "d", "e"]


def parSolution(N):
    for i in range(N):
        if goalList[i] != result[i]:
            return False
    return True


def Onblock(index, count):

    # break point of recursive call
    if count == len(goalList)+1:
        return True
    # copy tab of index value to result
    block = tab[index]
    # stack block
    result.append(block)
    print(result)
    if parSolution(count):
        print("Pushed a result solution ")
        # remove block from tab
        tab.remove(block)
        Onblock(0, count + 1)
    else:
        print("result solution not possible, back to the tab")
        # pop out if no partial solution
        result.pop()
        Onblock(index+1, count)


def Ontab(problem):
    # check if everything in stack is on the tab
    if len(problem) != 0:
        tab.append(problem.pop())
        Ontab(problem)
    # if everything is on the tab the we return true
    else:
        return True


def
    goal_stack_planing(problem)
    : # pop problem and put in
    tab Ontab(problem)
    # print index and number of blocks on result stack
    if Onblock(0, 1):
        print(result)

problem = ["c", "a", "e", "d", "b"]
print("Goal Problem")
for k, j in zip(goalList, problem):
    print(k+"       "+j)
print("\n")
goal_stack_planing(problem)
print("\nResult Solution:",result)
```

Output:

Name: Vikas Mane

Roll No: B1921152
Class: BE-B

Assignment 12

Title: Design and implement parallel algorithm utilizing all resources available. For

Binary Search for Sorted Array

Depth-First Search (tree or an undirected graph) OR

Breadth-First Search (tree or an undirected graph) OR

Best-First Search that (traversal of graph to reach a target in the shortest possible path)

System Requirements: C++ / Java

Theory:

In computer science, binary search, also known as half-interval search,logarithmic search,or binary chop,is a search algorithm that finds the position of a target value within a sorted array. Binary search compares the target value to the middle element of the array. If they are not equal, the half in which the target cannot lie is eliminated and the search continues on the remaining half, again taking the middle element to compare to the target value, and repeating this until the target value is found. If the search ends with the remaining half being empty, the target is not in the array. Even though the idea is simple, implementing binary search correctly requires attention to some subtleties about its exit conditions and midpoint calculation.

Binary search runs in logarithmic time in the worst case, making $O(\log n)$ comparisons,. Binary search is faster than linear search except for small arrays. However, the array must be sorted first to be able to apply binary search.

There are specialized data structures designed for fast searching, such as hash tables, that can be searched more efficiently than binary search. However, binary search can be used to solve a wider range of problems, such as finding the next-smallest or next-largest element in the array relative to the target even if it is absent from the array.

There are numerous variations of binary search. In particular, fractional cascading speeds up binary searches for the same value in multiple arrays. Fractional cascading efficiently solves a number of search problems in computational geometry and in numerous other fields. Exponential search extends binary search to unbounded lists. The binary search tree and B-tree data structures are based on binary search.

Objective: To study and implementation of parallel searching techniques.

Input: Sorted Array Numbers

Output: The key found at respective position/ Not found message

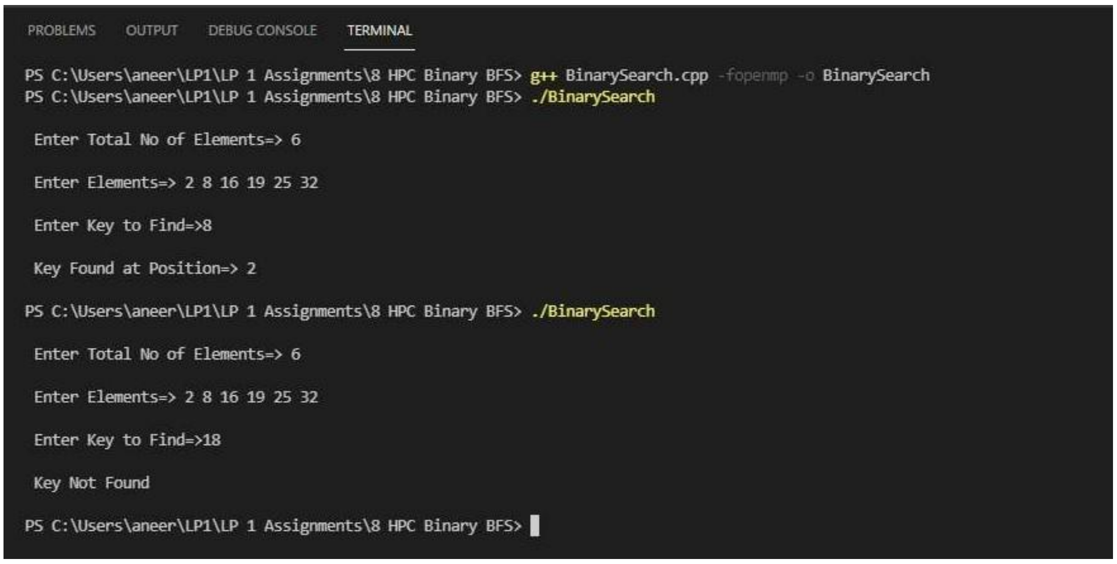
Result / Conclusion:
Hence, after the execution of the assignment, parallel searching is being implemented successfully

Program for Binary Search:

```
#include<iostream>
#include<stdlib.h>
#include<omp.h>
using namespace std;

int binary(int *, int, int, int);
int binary(int *a, int low, int high, int key)
{
  int mid;
  mid=(low+high)/2;
  int low1,low2,high1,high2,mid1,mid2,found=0,loc=-1;

  #pragma omp parallel sections
  {
      #pragma omp section
            {
            low1=low;
            high1=mid;
            while(low1<=high1)
            {
                  if(!(key>=a[low1] && key<=a[high1]))
                  {
                        low1=low1+high1;
                        continue;
                  }
                  mid1=(low1+high1)/2;
                  if(key==a[mid1])
                  {
                        found=1;
                        loc=mid1;
                        low1=high1+1;
                  }
                  else if(key>a[mid1])
                  {
                        low1=mid1+1;
                  }
                  else if(key<a[mid1])
                        high1=mid1-1;
            }
      }
          #pragma omp section
            {
                        low2=mid+1;
            high2=high;
            while(low2<=high2)
            {
                  if(!(key>=a[low2] && key<=a[high2]))
                  {
                        low2=low2+high2;
                        continue;
                  }
                  mid2=(low2+high2)/2;
                  if(key==a[mid2])
                  {
                        found=1;
                        loc=mid2;
                        low2=high2+1;
                  }
                  else if(key>a[mid2])
                  {
                  low2=mid2+1;
                  }
                  else if(key<a[mid2])
                  high2=mid2-1;
            }
```

```
        }
    }
    return loc;
}
int main()
{
    int *a,i,n,key,loc=-1;
    cout<<"\n Enter Total No of Elements=> ";
    cin>>n;
    a=new int[n];

    cout<<"\n Enter Elements=> ";
    for(i=0;i<n;i++)
      cin>>a[i];

    cout<<"\n Enter Key to Find=>";
    cin>>key;

    loc=binary(a,0,n-1,key);

    if(loc==-1)
        cout<<"\n Key Not Found\n\n";
    else
        cout<<"\n Key Found at Position=> "<<loc+1<<"\n\n";

    return 0;
}
```

Output for Binary Search:

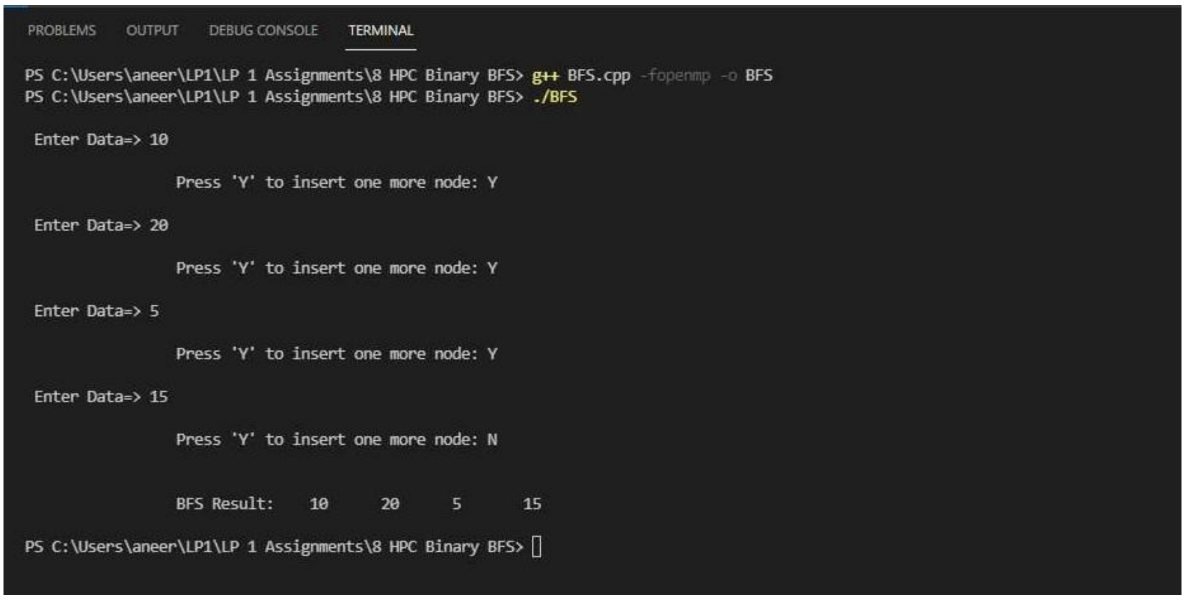


```
PS C:\Users\aneer\LP1\LP 1 Assignments\8 HPC Binary BFS> g++ BinarySearch.cpp -fopenmp -o BinarySearch
PS C:\Users\aneer\LP1\LP 1 Assignments\8 HPC Binary BFS> ./BinarySearch

 Enter Total No of Elements=> 6

 Enter Elements=> 2 8 16 19 25 32

 Enter Key to Find=>8

 Key Found at Position=> 2

PS C:\Users\aneer\LP1\LP 1 Assignments\8 HPC Binary BFS> ./BinarySearch

 Enter Total No of Elements=> 6

 Enter Elements=> 2 8 16 19 25 32

 Enter Key to Find=>18

 Key Not Found

PS C:\Users\aneer\LP1\LP 1 Assignments\8 HPC Binary BFS> 
```

Program for BFS:

```
#include<iostream>
#include<stdlib.h>
#include<queue>
#include<omp.h>
using namespace std;

class node
{
  public:
  node *left, *right;
  int data;
};
class Breadthfs
{
  public:
  node *insert(node *, int);
  void bfs(node *);
};
node *insert(node *root, int data)
{
  if(!root)
  {
      root=new node;
      root->left=NULL;
      root->right=NULL;
      root->data=data;
      return root;
  }
  queue<node *> q;
  q.push(root);
  while(!q.empty())
  {
      node *temp=q.front();
      q.pop();
      if(temp->left==NULL)
      {
            temp->left=new node;
            temp->left->left=NULL;
            temp->left->right=NULL;
            temp->left->data=data;
            return root;
      }
      else
            q.push(temp->left);
      if(temp->right==NULL)
      {
            temp->right=new node;
            temp->right->left=NULL;
            temp->right->right=NULL;
            temp->right->data=data;
            return root;
      }
      else
      {
      q.push(temp->right);
      }
  }
  return NULL;
}
void bfs(node *head)
{
      queue<node*> q;
      q.push(head);
      int qSize;
      while (!q.empty())
      {
             qSize = q.size();
             #pragma omp parallel for
             for (int i = 0; i < qSize; i++)
             {
                    node* currNode;
                    #pragma omp critical
                    {
                      currNode = q.front();
                      q.pop(); cout<<"\
                      t"<<currNode->data;
                    }
                    #pragma omp critical
                    {
                    if(currNode->left)
                           q.push(currNode->left);
                    if(currNode->right)
                           q.push(currNode->right);
                    }
             }
      }
}
int main(){
  node *root=NULL;
  int data;
  char ans;
  do
  {
      cout<<"\n Enter Data=> ";
      cin>>data;
      root=insert(root,data);
      cout<<"\n\t\t Press 'Y' to insert one more node: ";
      cin>>ans;
  }while(ans=='y'||ans=='Y');
  cout<<"\n";
  cout<<"\n\t\t BFS Result: ";
  bfs(root);
  cout<<"\n\n";
  return 0;
}
```

Output for BFS:



```
PS C:\Users\aneer\LP1\LP 1 Assignments\8 HPC Binary BFS> g++ BFS.cpp -fopenmp -o BFS
PS C:\Users\aneer\LP1\LP 1 Assignments\8 HPC Binary BFS> ./BFS

 Enter Data=> 10

                Press 'Y' to insert one more node: Y

 Enter Data=> 20

                Press 'Y' to insert one more node: Y

 Enter Data=> 5

                Press 'Y' to insert one more node: Y

 Enter Data=> 15

                Press 'Y' to insert one more node: N


                BFS Result:    10      20      5       15

PS C:\Users\aneer\LP1\LP 1 Assignments\8 HPC Binary BFS> 
```

Name: Vikas Mane

Roll No: B1921152

Class: BE-B

Virtual Lab Assignment

Problem Statement: Introduction to Fundamental of Fuzzy Logic and Basic Operations

Objective: To perform Fuzzy Logic Operations such as Addition, Subtraction, Complement, Union & Intersection

Theory:

Fuzzy Sets:

- A fuzzy set is a pair (A,m) where A is a set and m : A -->[0,1].
- For each x ε A, m(x) is called the grade of membership of x in (A,m). For a finite set A = {x1,...,xn}, the fuzzy set (A,m) is often denoted by {m(x1) / x1,...,m(xn) / xn}.
- Let x ε A. Then x is called not included in the fuzzy set (A,m) if m(x) = 0, x is called fully included if m(x) = 1, and x is called a fuzzy member if 0 < m(x) < 1.The set { x ε A | m(x) > 0 } is called the support of (A,m) and the set { x ε A | m(x) =1 } is called its kernel.

Fuzzy Set Operations:

Fuzzy Addition

Let us consider A1 = [a,b] and A2 = [c,d]

The addition of A1 and A2 is: [a,b] + [c,d] = [a+c, b+d]

Fuzzy Subtraction

Let us consider A1 = [a,b] and A2 = [c,d]

The subtraction of A1 and A2 is: [a,b] - [c,d] = [a-d, b-c]

Fuzzy Complement

The degree to which you believe something is not in the set is 1.0 minus the degree to which you believe it is in the set.

Fuzzy Intersection

If you have x degree of faith in statement A, and y degree of faith in statement B, how much faith do you have in the statement A and B?

Eg: How much faith in "that person is about 6' high and tall"

Fuzzy Union

If you have x degree of faith in statement A, and y degree of faith in statement B, how much faith do you have in the statement A or B?

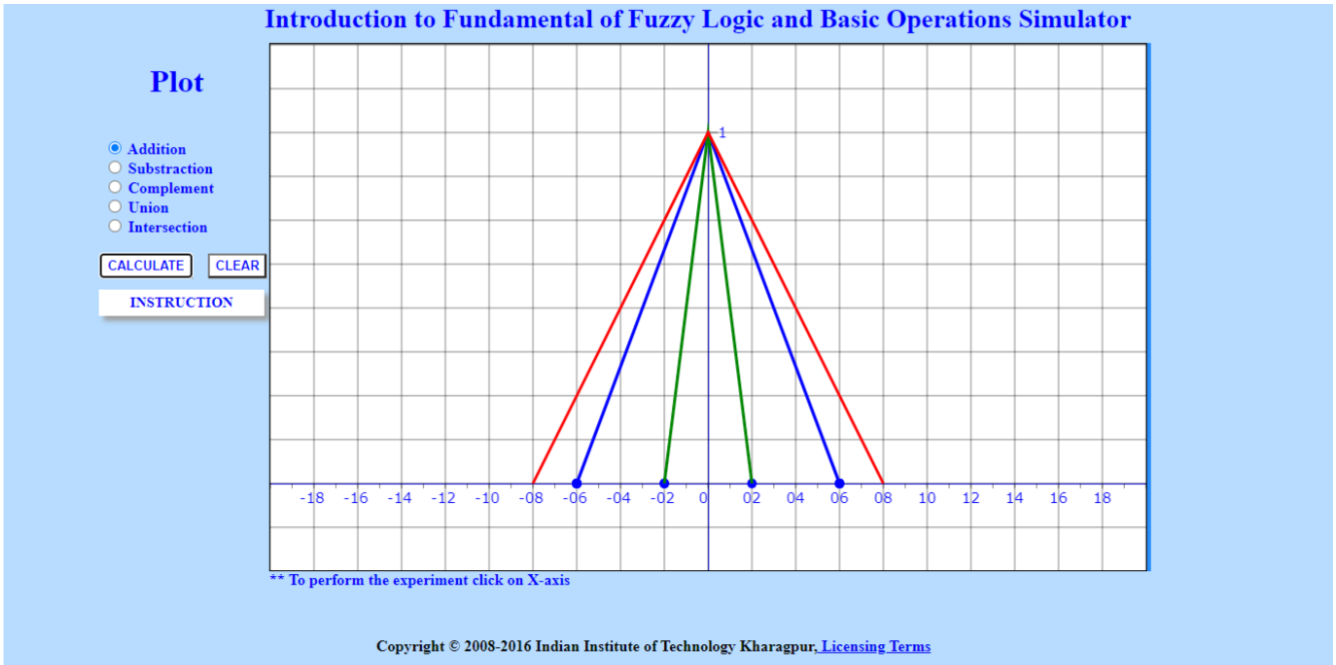
Eg: How much faith in "that person is about 6' high or tall"

Operations Executed:

1. Fuzzy Addition
2. Fuzzy Subtraction
3. Fuzzy Complement
4. Fuzzy Intersection
5. Fuzzy Union

Screenshot:

Conclusion: We have studied fundamentals of Fuzzy Logic and performed logical operations on the simulator.



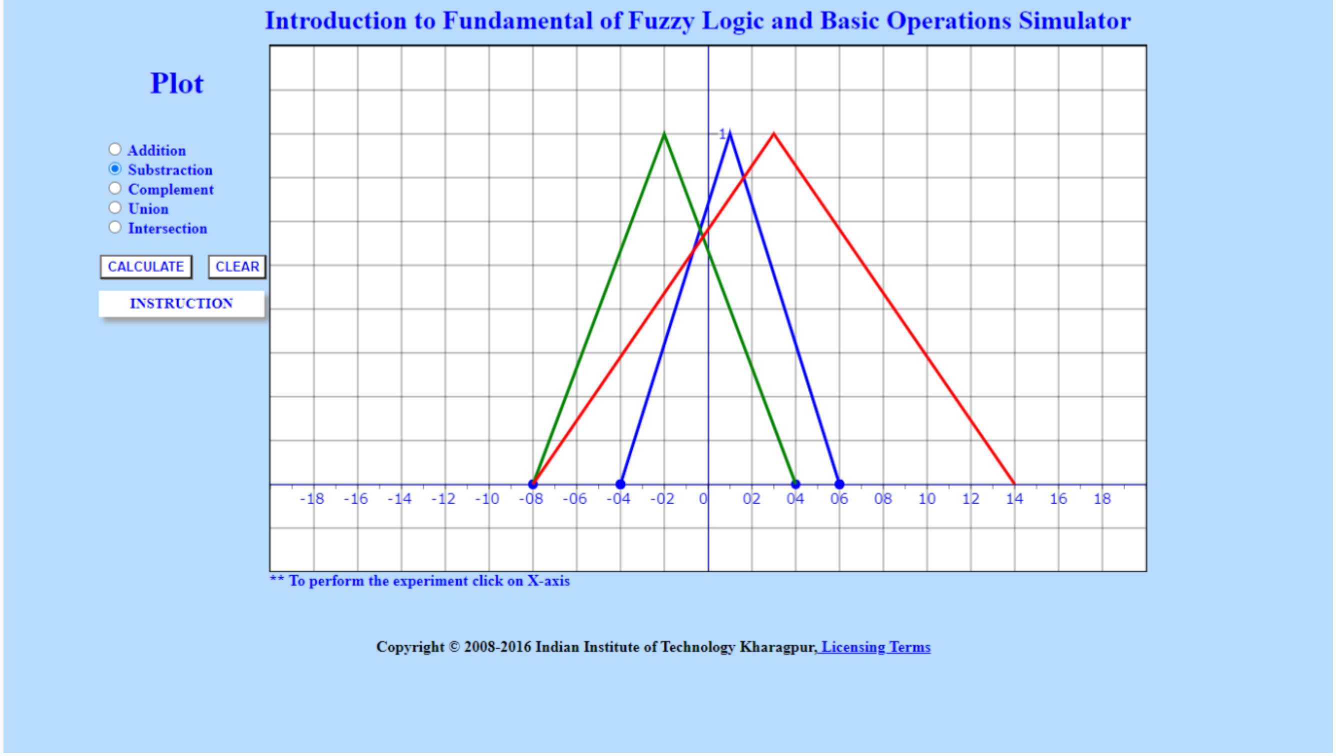
Introduction to Fundamental of Fuzzy Logic and Basic Operations Simulator

Plot

[✓ Addition]
Substraction
Complement
Union
Intersection

** To perform the experiment click on X-axis

Copyright © 2008-2016 Indian Institute of Technology Kharagpur, Licensing Terms



Introduction to Fundamental of Fuzzy Logic and Basic Operations Simulator

Plot

Addition
[✓ Substraction]
Complement
Union
Intersection

** To perform the experiment click on X-axis

Copyright © 2008-2016 Indian Institute of Technology Kharagpur, Licensing Terms