EXPERIMENT NO. 8
CURE

NAME:Rishabh Patil
SAP: 60009200056
BATCH: D12

AIM: Implement program for CURE Algorithm.

THEORY:

The CURE (Clustering Using Representatives) algorithm is a clustering method used in data mining and machine learning for partitioning datasets into groups or clusters. It's particularly useful for datasets where traditional methods like K-means may struggle due to irregular shapes or varying densities of clusters.

Here's an overview of how the CURE algorithm works:

1. Selection of Representative Points: Initially, a subset of points is chosen as representative points. These points aim to capture the overall characteristics of the dataset. Common approaches include random selection, sampling, or specific strategies to cover diverse areas of the dataset.

2. Hierarchical Clustering: The chosen representative points are clustered using a hierarchical clustering algorithm such as single-linkage or complete-linkage clustering. This step creates a hierarchical structure of clusters.

3. Pruning: The hierarchical structure might contain too many clusters or be too detailed. Pruning involves selecting a specified number of clusters to represent the dataset effectively. This can be achieved by cutting the hierarchical tree at a certain level or by merging clusters based on certain criteria.

4. Representative Points Update: After pruning, representative points for the selected clusters are updated to better represent their respective clusters. These updated representative points capture the essential characteristics of the clusters they represent.

5. Final Clustering: The final clusters are formed based on the updated representative points. Points in the dataset are assigned to the nearest representative point or cluster centroid, resulting in the final clustering of the dataset.

The key idea behind CURE is to use representative points to summarize clusters and facilitate clustering of large and complex datasets. It's advantageous for handling datasets with varying shapes, sizes, and densities of clusters. The hierarchical approach allows for flexibility in determining the granularity of clustering and enables the algorithm to capture diverse structures within the dataset.

However, CURE's performance might be sensitive to the choice of initial representative points and parameters used in the clustering process, such as the number of clusters to be formed and the pruning strategy.

Overall, CURE is effective in handling datasets where traditional clustering algorithms struggle, providing a robust approach to clustering data with varying characteristics and structures.

CODE WITH OUTPUT:

```
import numpy as np
import datetime

def comparator(x, y):
    if x[0] < y[0]:
        return -1
    if x[0] == y[0]:
        if x[1] < y[1]:
            return -1
        if x[1] > y[1]:
            return 1
        return 0
    return 1

ratio = 0.2
data_list = []
n = 0
sum = 0
k = 3
start = datetime.datetime.now()

# Hierarchy clustering
print("Enter data in the format 'x,y', or 'q' to quit.")
while True:
    user_input = input("Enter data (x,y): ")
    if user_input.lower() == 'q':
        break
    values = user_input.strip().split(',')
    if len(values) != 2:
        print("Invalid input. Please enter data in the format 'x,y'.")
        continue
    try:
        x, y = float(values[0]), float(values[1])
        data_list.append([[x, y], 1, [[x, y]]])
        n += 1
    except ValueError:
```

```
        print("Invalid input. Please enter numeric values for x and
y.")

num = 1
while n != 3:
    min_coordinate = None
    min_distance = float('inf')
    num += 1
    for i in range(len(data_list)):
        cluster1 = data_list[i]
        centroid1 = np.array(cluster1[0]) / cluster1[1]
        for j in range(len(data_list)):
            if j > i:
                cluster2 = data_list[j]
                centroid2 = np.array(cluster2[0]) / cluster2[1]

                distance = (centroid2 - centroid1) ** 2
                if np.sum(distance) < min_distance:
                    min_coordinate = [i, j]
                    min_distance = np.sum(distance)

    cor1 = data_list[min_coordinate[0]]
    cor2 = data_list[min_coordinate[1]]
    new_cluster = [np.ndarray.tolist(np.array(cor1[0]) +
np.array(cor2[0]), cor1[1] + cor2[1], cor1[2] + cor2[2])]
    data_list.remove(cor1)
    data_list.remove(cor2)
    data_list.append(new_cluster)

    n -= 1

centroid = []
# Find representatives and calculate centroids
id = 0
re = []
for cluster in data_list:
    sorted_cluster = sorted(cluster[2], key=lambda x: (x[0], x[1]))
    repre = [sorted_cluster[0]]
    iter = 0
    while iter < n:
        iter += 1
        candi = None
        min_distance = float('inf')
        if len(repre) == 1:
            center = np.array(repre)
        else:
            center = np.sum(np.array(repre), axis=0) / len(repre)
```

```
        for coor in sorted_cluster:
            if coor not in repre:
                dis = np.sum((center - np.array(coor)) ** 2)
                if dis < min_distance:
                    min_distance = dis
                    candi = coor
        repre.append(candi)
    clustercentroid = np.sum(np.array(cluster[2]), axis=0) /
len(cluster[2])
    centroid.append(clustercentroid)
    l = []
    for representative in repre:
        representative = np.ndarray.tolist(np.array(representative) +
ratio * (clustercentroid - np.array(representative)))
        l.append(representative)
    re.append(l)

# CURE
print("Enter data to process in the format 'x,y', or 'q' to quit.")
while True:
    user_input = input("Enter data (x,y): ")
    if user_input.lower() == 'q':
        break
    values = user_input.strip().split(',')
    if len(values) != 2:
        print("Invalid input. Please enter data in the format 'x,y'.")
        continue
    try:
        point = [float(values[0]), float(values[1])]
        min_distance = float('inf')
        index_cluster = 0
        for i in range(len(re)):
            cluster = re[i]
            for r in range(len(cluster)):
                representative = cluster[r]
                dis = np.sum((np.array(representative) -
np.array(point)) ** 2)
                if dis < min_distance:
                    min_distance = dis
                    index_cluster = i

        data_list[i][2].append(point)
    except ValueError:
        print("Invalid input. Please enter numeric values for x and
y.")

for x in range(len(data_list)):
```

```
    print(data_list[x][2])
```

Output:

```
Enter data in the format 'x,y', or 'q' to quit.
Enter data (x,y): (5, 8), (23, 18), (7, 14), (15, 21), (9, 12), (32, 11), (6, 25),
Invalid input. Please enter data in the format 'x,y'.
Enter data (x,y): (5, 8)
Invalid input. Please enter numeric values for x and y.
Enter data (x,y): 5
Invalid input. Please enter data in the format 'x,y'.
Enter data (x,y): 5,10
Enter data (x,y): 12,30
Enter data (x,y): 51,78
Enter data (x,y): 45,35
```

CONCLUSION:
We have successfully implemented program for CURE Algorithm.