Advanced Algorithm
Lab
Final Assessment

19MIC0016
Pranauv.A.J

Question 1:
Given a multiset of positive integers (multiset is a set where elements can be repeated) whose cardinality is divisible by 3, partition the given set into triplets (i.e., three elements for a set) such that sum of the elements of each partition is the same as the others (i.e., all partitions are having equal sum). Note that every element of the multiset must be used exactly once in a partition. For a given multiset of positive integers (ask the user to provide the input and check whether it meets the initial condition), write a program using backtracking method to find the solution for this problem (Hint: divide the elements into triplets, check the total with the total of another triplet set. If equal, check with another and continue. If not equal for two triplet sets, redo the partition).

Code:

```
#include <iostream>
#include <numeric>
using namespace std;
void part(int S[], int n, int k);
bool check(int sumLeft[], int k);

bool s_sum(int S[], int n, int sumLeft[], int
A[], int k)
{
    if (check(sumLeft, k)) {
        return true;
    }
     if (n < 0) {
        return false;
    }
    bool result = false;
    for (int i = 0; i < k; i++)
    {
        if (!result && (sumLeft[i] - S[n]) >=
0)
        {

            A[n] = i + 1;
            sumLeft[i] = sumLeft[i] - S[n];
            result = s_sum(S, n - 1, sumLeft,
A, k);
            sumLeft[i] = sumLeft[i] + S[n];
        }
    }
```

```
    return result;
}



int main()
{
    cout<<"19MIC0016\nPranauv.A.J\n\n";
    int n;
    cout<<"Enter the number of elements in the
multiset: ";
    cin>>n;
    int S[n];
    cout<<"\nEnter the elements in the
multiset\n";
    for(int i=0;i<n;i++)
        cin>>S[i];
    int k = n/3;

    part(S, n, k);

    return 0;
}

void part(int S[], int n, int k)
{

    if (n < k)
    {
        cout << "k-part of set S is not
```

```
possible";
        return;
    }


    int sum = accumulate(S, S + n, 0);

    int A[n], sumLeft[k];


    for (int i = 0; i < k; i++) {
        sumLeft[i] = sum/k;
    }

    bool result = !(sum % k) && s_sum(S, n - 1,
sumLeft, A, k);

    if (!result)
    {
        cout << "k-part of set S is not
possible";
        return;
    }

    for (int i = 0; i < k; i++)
    {
        cout << "part "<< i+1<<" ";
        for (int j = 0; j < n; j++)
        {
            if (A[j] == i + 1) {
```

```
                cout << S[j] << " ";
            }
        }
        cout << endl;
    }
}

bool check(int sumLeft[], int k)
{
    int r = true;
    for (int i = 0; i < k; i++)
    {
        if (sumLeft[i] != 0) {
            r = false;
        }
    }

    return r;
}
```

Output:

```
19MIC0016
Pranauv.A.J

Enter the number of elements in the multiset: 12

Enter the elements in the multiset
20
23
25
30
45
45
27
30
30
40
22
23
Partition 1 45 22 23
Partition 2 23 27 40
Partition 3 30 30 30
Partition 4 20 25 45
```

Question 2:

Given a graph, consider a property for a subset V' of V: no two vertices of V' are adjacent. The questions to you are the following: (i) Given a graph (say in adjacency matrix) and subset V', write a program that checks this property is hold for V' or not. (ii) Write a program that proposes an approximation algorithm and outputs V' with the above property.

Code:

```
#include<stdio.h>
#include<stdlib.h>
int arr[100][100];
int cad[100];
int blue[100];
int red[100];
int main()
{
    int n;
    int m;
    int b;
    int r;
    printf("19MIC0016\nPranauv.A.J\n\n");
    printf("Enter the number of vertices:");
    scanf("%d",&n);
    printf("Enter the adjacency matrix:");
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<n;j++)
        {
            scanf("%d",&arr[i][j]);
```

```
        }
    }
    printf("Enter the number of elements in V:");
    scanf("%d",&m);
    printf("Enter the elements of V:");
    for(int i=0;i<m;i++)
    {
        scanf("%d",&cad[i]);
    }
    printf("Enter the number of elements in set of
blue vertices:");
    scanf("%d",&b);
    printf("Enter the elements in blue set:");
    for(int i=0;i<b;i++)
    {
        scanf("%d",&blue[i]);
    }
    printf("Enter the number of elements in set of
red vertices:");
    scanf("%d",&r);
    printf("Enter the elements in red set:");
    for(int i=0;i<r;i++)
    {
        scanf("%d",&red[i]);
    }
    for(int i=0;i<m;i++)
    {
        for(int j=0;j<n;j++)
        {
            if(arr[cad[i]-1][cad[i+1]-1]==1)
            {
```

```
                printf("It should not have
adjacent edges");
                exit(0);
            }
        }
    }
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<n;j++)
        {
            if(cad[i]==blue[i])
            {
                for(int k=0;k<r;k++)
                {
                    if(arr[cad[i]-1][k]==1)
                    {
                        for(int k1=0;k1<r;k1++)
                        {

if(arr[cad[i]-1][k]==blue[k1])
                            {
                                printf("Connection
should not be between same sets:");
                                exit(0);
                            }
                        }
                    }
                }
            }
            if(cad[i]==red[i])
            {
```

```
                for(int k=0;k<r;k++)
                {
                    if(arr[cad[i]-1][k]==1)
                    {
                        for(int k1=0;k1<r;k1++)
                        {

if(arr[cad[i]-1][k]==red[k1])
                                {
                                    printf("Connection
should not be between same sets:");
                                    exit(0);
                                }
                            }
                    }
                }
            }
        }
    }
    printf("THE GIVEN SUBSET IS A APRROXIMATION
SOLUTION");
    for(int i=0;i<m;i++)
    {
        printf("%d\t",cad[i]);
    }
}
```

Output:

```
19MIC0016
Pranauv.A.J

Enter the number of vertices:8
Enter the adjacency matrix:
0 1 1 0 1 0 0 0
1 0 0 1 0 1 0 0
1 0 0 1 0 0 1 0
0 1 1 0 1 0 0 0
1 0 0 0 0 1 1 0
0 1 0 0 1 0 0 1
0 0 1 0 1 0 0 1
0 0 0 1 0 1 1 0
Enter the number of elements in V:4
Enter the elements of V:2 3 5 6
Enter the number of elements in set of blue vertices:4
Enter the elements in blue set:2 3 5 6
Enter the number of elements in set of red vertices:4
Enter the elements in red set:1 4 7 8
It should not have adjacent edges
```