# Question 1:

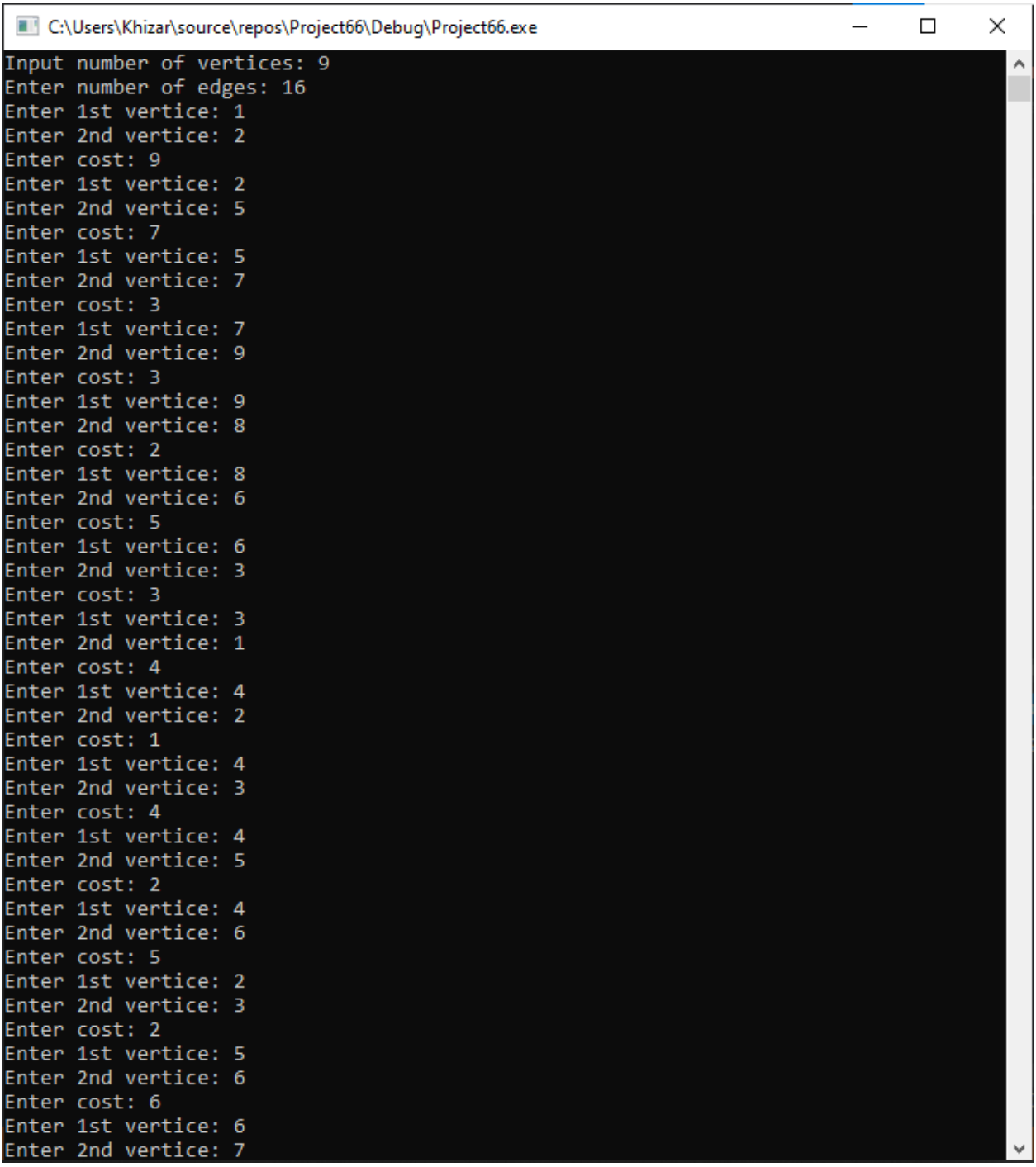
```
#include <iostream>
using namespace std;


void printMST(int** matrix, int* parent, int n)//function for printing MST
{
	int cost= 0;
	cout << "edges	cost" << endl;
	for (int i = 1; i < n; i++)
	{		//vertex 1		vertex 2		weight
		cout << parent[i] << "--" << i << "	" << matrix[i][parent[i]] << endl;
		cost += matrix[i][parent[i]];//adding total cost
	}
	cout << "Total Cost: " << cost;
}
void createMST(int** matrix, int n)//function to create MST
{
	int* edges; int* parent; int* visit;
	edges = new int[n];//array of edges
	parent = new int[n];//array for parent
	visit = new int[n];// array for checking visited vertex
	int temp = 100, index;
	for (int i = 0; i < n; i++)//initializing all the arrays
	{
		edges[i] = 100;
		visit[i] = 0;
		parent[i] = -1;
	}
	edges[0] = 0;

	for (int i = 0; i < n-1; i++)
	{
		temp = 100;
		for (int j = 0; j < n; j++)
		{
			if (visit[j] == 0 && edges[j] < temp)//chacking for minimum cost
edge adjacent to parent
			{
				index = j;
				temp = edges[j];
			}
		}
		visit[index] = 1;//checking the vertex for visit
		for (int k = 0; k < n; k++)
		{
			if (matrix[index][k] != 0 && visit[k] == 0 && matrix[index][k] <
edges[k] )//condition for getting the weight between both vertices
			{
				edges[k] = matrix[index][k];//minmum edge corresponding to k
vertex
				parent[k] = index;//parent of k vertex
			}
		}
	}
	printMST(matrix, parent, n);//funtion to print MST
```

```
}
int main()
{
	int** arr;
	int n, m, c;
	cout << "Input number of vertices: ";//getting number of vertices
	cin >> n;
	arr = new int* [n];//matrix of certain vertices
	for (int i = 0; i < n; i++)//making matrix
	{
		arr[i] = new int[n];
	}
	for (int i = 0; i < n; i++)//initializing matrix with 0
	{
		for (int j = 0; j < n; j++)
		{
			arr[i][j] = 0;
		}
	}
	cout << "Enter number of edges: ";//get number of edges
	cin >> m;
	int x, y;
	for (int k = 0; k < m; k++)
	{
		cout << "Enter 1st vertice: ";//1st vertice
		cin >> x;
		cout << "Enter 2nd vertice: ";//2nd vertice
		cin >> y;
		cout << "Enter cost: ";//cost between two vertices
		cin >> c;
		arr[x - 1][y - 1] = c;
		arr[y - 1][x - 1] = c;
	}
	cout << "Adjacency Matrix: " << endl << "   ";
	for (int i = 0; i < n; i++)//adcency matrix making
	{
		cout<< i + 1 << " ";
	}
	cout << endl;
	for (int i = 0; i < n; i++)//output adjacency matrix
	{
		cout << i + 1 << " ";
		for (int j = 0; j < n; j++)
		{
			cout << arr[i][j] << " ";
		}
		cout << endl;
	}
	createMST(arr, n);//function to create MST
	system("pause");
	return 0;
}
```

## Output:


C:\Users\Khizar\source\repos\Project66\Debug\Project66.exe
Input number of vertices: 9
Enter number of edges: 16
Enter 1st vertice: 1
Enter 2nd vertice: 2
Enter cost: 9
Enter 1st vertice: 2
Enter 2nd vertice: 5
Enter cost: 7
Enter 1st vertice: 5
Enter 2nd vertice: 7
Enter cost: 3
Enter 1st vertice: 7
Enter 2nd vertice: 9
Enter cost: 3
Enter 1st vertice: 9
Enter 2nd vertice: 8
Enter cost: 2
Enter 1st vertice: 8
Enter 2nd vertice: 6
Enter cost: 5
Enter 1st vertice: 6
Enter 2nd vertice: 3
Enter cost: 3
Enter 1st vertice: 3
Enter 2nd vertice: 1
Enter cost: 4
Enter 1st vertice: 4
Enter 2nd vertice: 2
Enter cost: 1
Enter 1st vertice: 4
Enter 2nd vertice: 3
Enter cost: 4
Enter 1st vertice: 4
Enter 2nd vertice: 5
Enter cost: 2
Enter 1st vertice: 4
Enter 2nd vertice: 6
Enter cost: 5
Enter 1st vertice: 2
Enter 2nd vertice: 3
Enter cost: 2
Enter 1st vertice: 5
Enter 2nd vertice: 6
Enter cost: 6
Enter 1st vertice: 6
Enter 2nd vertice: 7

```
Enter cost: 8
Enter 1st vertice: 7
Enter 2nd vertice: 8
Enter cost: 1
Adjacency Matrix:
  1 2 3 4 5 6 7 8 9
1 0 9 4 0 0 0 0 0 0
2 9 0 2 1 7 0 0 0 0
3 4 2 0 4 0 3 0 0 0
4 0 1 4 0 2 5 0 0 0
5 0 7 0 2 0 6 3 0 0
6 0 0 3 5 6 0 8 5 0
7 0 0 0 0 3 8 0 1 3
8 0 0 0 0 0 5 1 0 2
9 0 0 0 0 0 0 3 2 0
edges     cost
3--2      2
1--3      4
2--4      1
4--5      2
3--6      3
5--7      3
7--8      1
8--9      2
Total Cost: 18Press any key to continue . . .
```

# Question 2:

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


void printMST(int** matrix, int* parent, int n)
{
	int minCost = 0;//getting minimum cost
	cout << "edges   cost" << endl;
	for (int i = 0; i < n; i++)
	{

		if (parent[i] != -1)
		{  //       vertex 1         vertex 2        cost between them
			cout << parent[i] + 1 << "--" << i + 1<< "    " <<
matrix[parent[i]][i] << endl;
			minCost += matrix[parent[i]][i];//adding minimum cost
		}

	}
	cout << "minimum cost: " << minCost << endl;
}
void createMST(int** matrix, int n, int* cost, int m)//function to create MST
{
	int* parent;
	int* parent1;
	parent1 = new int[n];
	parent = new int[n];//array for getting vertices
	bool* ok;
	ok = new bool[m];//checking if the weight is visited
	int edge_check = 0;//checking no of edges
	for (int i = 0; i < n; i++)
	{
		parent[i] = -1;
	}

	for (int i = 0; i < m; i++)
	{
		ok[i] = false;
	}

	for (int i = 0; i < m; i++)//loop for checking weight
	{
		for (int j = 0; j < n; j++)
		{
			for (int k = 0; k < n; k++)
			{
				if (matrix[j][k] == cost[i] && ok[i] == false && edge_check <
n-1)//condition to get the weight
				{

					if (parent[j] != parent[k]  || parent[k]  == -1)
					{

						if (parent[k] == -1 && parent1[k] != parent[j])
						{
```

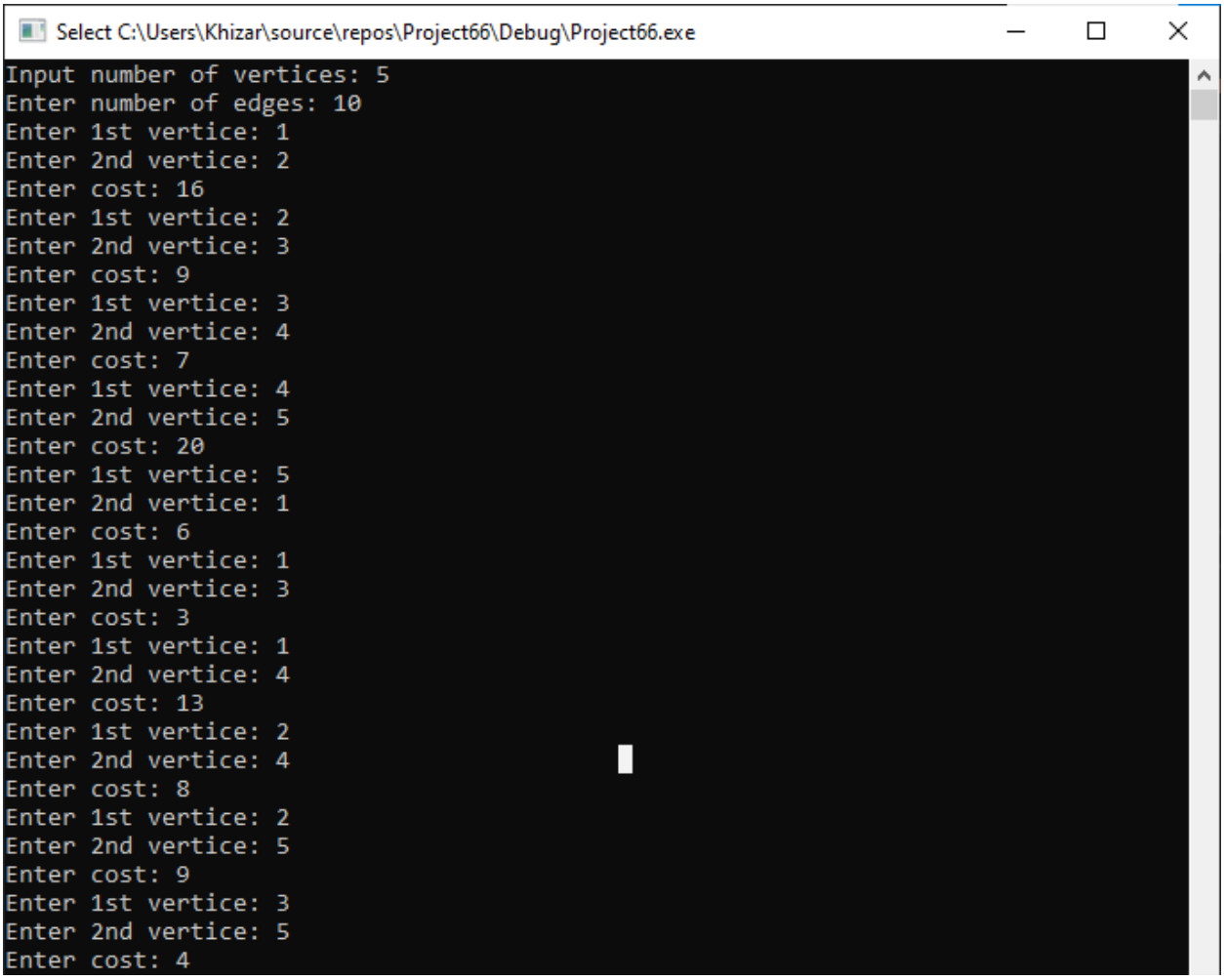
```
                                                ok[i] = true;
                                                parent[k] = j;//j is the parent of k
vertex
                                                parent1[j] = k;
                                                edge_check++;//no of edge created
                                        }

                                }

                        }
                }
        }
        cout << endl;
        printMST(matrix, parent, n);//function to print edges
}
int main()
{
        int** arr;
        int n, m, c;
        cout << "Input number of vertices: ";//getting number of vertices
        cin >> n;
        arr = new int* [n];//matrix of certain vertices
        for (int i = 0; i < n; i++)//making matrix
        {
                arr[i] = new int[n];
        }
        for (int i = 0; i < n; i++)//initializing matrix with 0
        {
                for (int j = 0; j < n; j++)
                {
                        arr[i][j] = 0;
                }
        }
        cout << "Enter number of edges: ";//get number of edges
        cin >> m;
        int* costs;
        costs = new int[m];//array for saving costs of all edges
        int x, y;
        for (int k = 0; k < m; k++)
        {
                cout << "Enter 1st vertice: ";//1st vertice
                cin >> x;
                cout << "Enter 2nd vertice: ";//2nd vertice
                cin >> y;
                cout << "Enter cost: ";//cost between two vertices
                cin >> c;
                arr[x - 1][y - 1] = c;
                arr[y - 1][x - 1] = c;
                costs[k] = c;
        }
        cout << "Adjacency Matrix: " << endl;
        cout << "   ";
        for (int i = 0; i < n; i++)//adjacency matrix making
        {
                cout<< i + 1 << " ";
        }
        cout << endl;
```

```
	for (int i = 0; i < n; i++)//output adjacency matrix
	{
		cout << i + 1 << " ";
		for (int j = 0; j < n; j++)
		{
			cout << arr[i][j] << " ";
		}
		cout << endl;
	}
	for (int i = 0; i < m; i++)
	{
		for (int j = 0; j < m; j++)//sorting the weights in increasing order
		{
			if (costs[j] > costs[j + 1] && j+1 < m)
			{
				int t = costs[j];
				costs[j] = costs[j + 1];
				costs[j + 1] = t;
			}
		}
	}
	cout << endl;
	createMST(arr, n, costs, m);//function to create MST
	system("pause");
	return 0;
}
```

## Output:


Select C:\Users\Khizar\source\repos\Project66\Debug\Project66.exe
Input number of vertices: 5
Enter number of edges: 10
Enter 1st vertice: 1
Enter 2nd vertice: 2
Enter cost: 16
Enter 1st vertice: 2
Enter 2nd vertice: 3
Enter cost: 9
Enter 1st vertice: 3
Enter 2nd vertice: 4
Enter cost: 7
Enter 1st vertice: 4
Enter 2nd vertice: 5
Enter cost: 20
Enter 1st vertice: 5
Enter 2nd vertice: 1
Enter cost: 6
Enter 1st vertice: 1
Enter 2nd vertice: 3
Enter cost: 3
Enter 1st vertice: 1
Enter 2nd vertice: 4
Enter cost: 13
Enter 1st vertice: 2
Enter 2nd vertice: 4
Enter cost: 8
Enter 1st vertice: 2
Enter 2nd vertice: 5
Enter cost: 9
Enter 1st vertice: 3
Enter 2nd vertice: 5
Enter cost: 4

```
Adjacency Matrix:
  1 2 3 4 5
1 0 16 3 13 6
2 16 0 9 8 9
3 3 9 0 7 4
4 13 8 7 0 20
5 6 9 4 20 0


edges   cost
4--2    8
1--3    3
3--4    7
3--5    4
minimum cost: 22
Press any key to continue . . .
```

Roll no :- 21F-4171 Section :- BSCS 3A

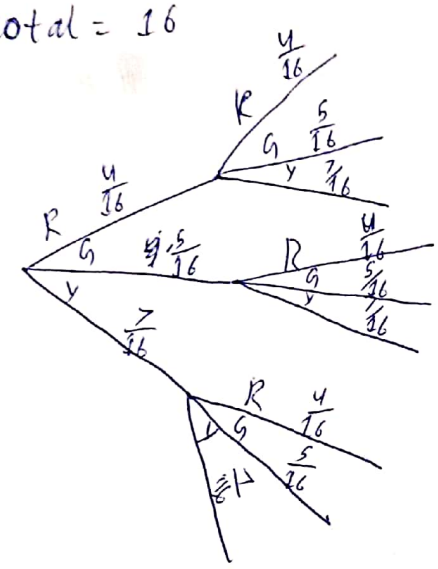
## Probability with Replacement

The probability in which we can take out one item and can Replace it with another item is called Probability with Replacement

**Example:-**

We have 4 red candies, 5 green candies and 7 yellow candies. Find the we draw 2 candies with Replacement. Find the Probability that both are Red Candies

total = 16




$$P = \frac{4}{16} \times \frac{4}{16} = \frac{1}{16}$$

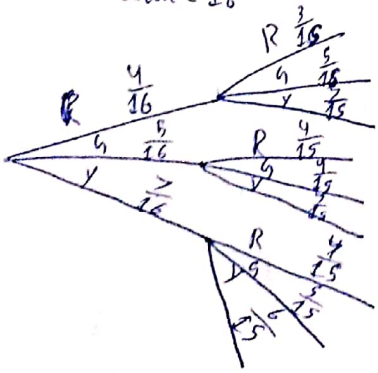
$\frac{1}{16}$ Probability to get both Red Candies

## Probability without Replacement

The probability in which we can't take out a item more than once is called Probability without Replacement.

**Example:-** We have 4 Red, 5 green and 7 yellow candies. we draw 2 candies without Replacement Find Probability that both are yellow.

total = 16




$$P = \frac{7}{16} \times \frac{6}{15} = \frac{42}{240} = \frac{7}{40}$$

$\frac{7}{40}$ Probability to get both yellow candies

## Mutually inclusive

The events which can occur together or have common outcome is called Mutually inclusive.

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

**Example:-**

take out a spade or jack from full deck of cards

A = Card is Spade
B = Card is Jack

$P(A) = \frac{13}{52}$, $P(B) = \frac{4}{52}$, $P(A \cap B) = \frac{1}{52}$ (common event of getting a spade jack)

$P(A \cup B) = \frac{13}{52} + \frac{4}{52} - \frac{1}{52} = \boxed{\frac{4}{13}}$ is the Probability of getting [redacted] spade or Ja

## Mutually Exclusive

The events which can't occur together or can't have a common outcome is called Mutually Exclusive.

$$P(A \cup B) = P(A) + P(B)$$

**Example:-**

take out a queen or king card from full deck of cards

A = Queen B = king

$P(A) = \frac{4}{52}$, $P(B) = \frac{4}{52}$

$P(A \cup B) = P(A) + P(B) = \frac{4}{52} + \frac{4}{52} = \frac{8}{52} = \frac{2}{13}$

## Baye's theorem

When the Probability of the event depends on the event which has happened we find it using Baye's theorem.

$$P(A|B) = \frac{P(A)\,P(B|A)}{P(B)}$$

**Example:**

A Couple has 2 children, older one is a Boy, Find if both are Boys

A = Both are Boys
B = older is Boy
$P(A) = \frac{1}{4}$, $P(B) \frac{1}{2}$

$$P(A|B) = \frac{P(A)\,P(B|A)}{P(B)} = \frac{\frac{1}{4} \cdot 1}{\frac{1}{2}} = \frac{1}{2}$$

Roll no:- 21F-9171 Section:- BSCS-3A

## Equally Likely events:-

The events which have equal chances of occuring ~~or~~ means have equal Probability is called Equally likely events.

**Example:-**

On a standard dice every number have equal chances of occuring and have $\frac{1}{6}$ Probability.

## Complement Rule

An Event B which has all the outcomes that do not occur in Event A is called Complement Rule.

**Example:**

What ~~are~~ is the probability that after rolling the dice the number is not 6.

Probability of getting 6 = $P(A) = \frac{1}{6}$

$P(B) = 1 - P(A)$

$= 1 - \frac{1}{6} = \frac{6-1}{6} = \frac{5}{6}$ Probability of 6 not occuring.

## Addition theorem

The Sum of Probability of two events that are not occuring at same time or does not have anything in common.

**Example**

What is the Probability of getting 1 or 6 when the dice is rolled.

A = getting 6

B = getting 1

$P(A) = \frac{1}{6}$ , $P(B) = \frac{1}{6}$

$P(A \cup B) = P(A) + P(B) = \frac{1}{6} + \frac{1}{6} = \frac{2}{6} = \frac{1}{3}$

## Multipulication theorem

The Product of Probability of two events that do not occur at same time and do not have anything common.

**Example:-** A dice is rolled twice, what is the probability of getting both 6.

A = Getting first 6
B = Getting second 6

$P(A) = \frac{1}{6}$, $P(B) = \frac{1}{6}$

$P(AB) = P(A)\,P(B)$
$= \frac{1}{6} \times \frac{1}{6} = \boxed{\frac{1}{36}}$

## Partition theorem

if we don't know probability of one event we can find it using probability of other event which is known.

## Example

Company A supply 80% Widgets in which 1% are defective and Company B supplies 20% widgets in which 3% are defective. if a guy randomly purchaces a widget. Find the Probability that it's defective.

$P(A) = 80\% = 0.8$
$P(B) = 20\% = 0.2$
$P(D) =$ Defective Probability

$P(D|A) = 0.01$
$P(D|B) = 0.03$

$P(D) = P(D|A) \cdot P(A) + P(D|B) \cdot P(B)$

$P(D) = (0.01)(0.8) + (0.03)(0.2)$

$\boxed{P(D) = 0.014}$ Probability that the Purchased widget is defective