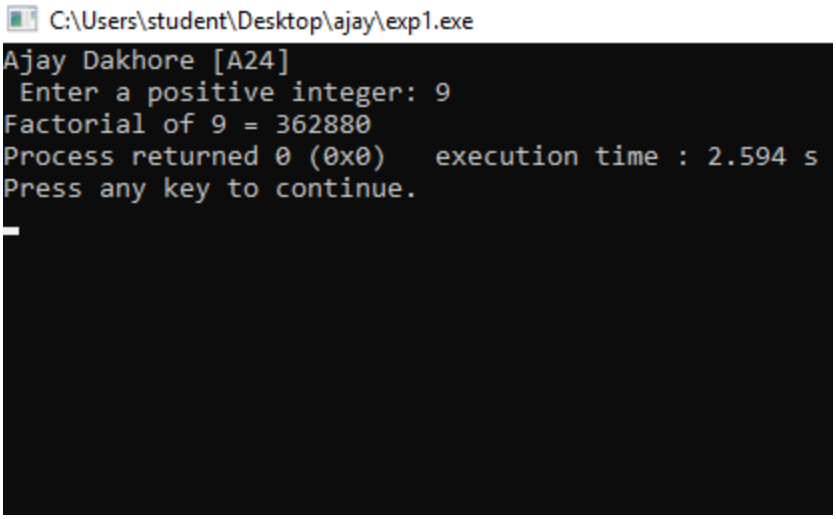
Experiment no.01

```
// A24) AJAY DAKHORE

//1)Factorial using recursive function

#include<iostream>

using namespace std;

int factorial(int n);

int main()

{

int n;

cout<<"Ajay Dakhore [A24]";

cout << "\n Enter a positive integer: ";

cin >> n;

cout << "Factorial of " << n << " = " << factorial(n);

return 0;

}

int factorial(int n)

{

if(n > 1)

return n * factorial(n - 1);

else

return 1;
```

}

output:




```
// A24) AJAY DAKHORE

//2) factorial using iterative function

#include <iostream>

using namespace std;

int fact_iter(int n)

{

int result = 1;

for (int i = 1; i <= n; i++)

{

result *= i;

}

return result;

}

int main()

{
```

```
int n;

cout<<"Ajay Dakhore [A24]";

while (1)

{

cout<<"\n Enter interger (0 to exit): ";

cin>>n;

if (n == 0)

break;

cout<<fact_iter(n)<<endl;

}

return 0;

}
```

output:

C:\Users\student\Desktop\exp11.exe

```
Ajay Dakhore [A24]
 Enter interger (0 to exit): 9
362880

 Enter interger (0 to exit): 2
2

 Enter interger (0 to exit): 6
720

 Enter interger (0 to exit):
```

Experiment no.02

```
// A24) AJAY DAKHORE
//C++ Program - Binary Search
#include<iostream>
using namespace std;
int main()
{
int n, i, arr[50], search, first, last, middle;
cout<<"Ajay Dakhore [A24]";
cout<<"\n Enter total number of elements :";
cin>>n;
cout<<"Enter "<<n<<" number :";
for (i=0; i<n; i++)
{
cin>>arr[i];
}
cout<<"Enter a number to find :";
cin>>search;
first = 0;
last = n-1;
middle = (first+last)/2;
while (first <= last)
{
```

```
if(arr[middle] < search)
{
first = middle + 1;


}
else if(arr[middle] == search)
{
cout<<search<<" found at location "<<middle+1<<"\n";
break;
}
else
{
last = middle - 1;
}
middle = (first + last)/2;
}
if(first > last)
{
cout<<"Not found! "<<search<<" is not present in the list.";
}
return 0;
}
```

output:

C:\Users\student\Desktop\exp2.exe

```
Ajay Dakhore [A24]
 Enter total number of elements :5
Enter 5 number :23 34 45 56 60
Enter a number to find :34
34 found at location 2

Process returned 0 (0x0)   execution time : 60.147 s
Press any key to continue.
```

```
// A24) AJAY DAKHORE

// binary search

#include <iostream>

using namespace std;

int bs(int a[],int l,int r,int x)

{

if(r>=l)

{

int m=l+(r-l)/2;

if(a[m]==x)

return m;

else if(a[m]>x)

return bs(a,l,m-1,x);

else

return bs(a,m+1,r,x);

}

return -1;

}
```

```
int main()
{
int a[10],n,i,key,res;
cout<<"Ajay Dakhore [A24]";


cout << "\nEnter size of array" << endl;
cin>>n;
cout<<"\nEnter the Array in Ascending order::\t";
for(i=0;i<n;i++)
{
cin>>a[i];
}
cout<<"\nEntered Array Is ::\t";
for(i=0;i<n;i++)
{
cout<<"\n";
cout<<a[i];
}
cout<<"\nEnter the element to be searched::\t";
cin>>key;
res=bs(a,0,n-1,key);
if(res==-1)
cout<<"\nElement not found ";
else
cout<<"\nElement found at position "<<res;
return 0;
}
```

output:

C:\Users\student\Desktop\exp22.exe

```
Ajay Dakhore [A24]
Enter size of array
5

Enter the Array in Ascending order::    23
45
56
67
78

Entered Array Is ::
23
45
56
67
78
Enter the element to be searched::      56

Element found at position 2
Process returned 0 (0x0)   execution time : 22.971 s
Press any key to continue.
```

Experiment no.03

```
// A24) AJAY DAKHORE

// quick sort

#include <iostream>

using namespace std;

void quick_sort(int[],int,int);

int partition(int[],int,int);

int main()

{

int a[50],n,i;

cout<<"Ajay Dakhore [A24]";

cout<<"\nHow many elements?";

cin>>n;

cout<<"\nEnter array elements:";

for(i=0;i<n;i++)

cin>>a[i];


quick_sort(a,0,n-1);

cout<<"\nArray after sorting:";

for(i=0;i<n;i++)

cout<<a[i]<<" ";

return 0;
```

```
}
void quick_sort(int a[],int l,int u)
{
int j;
if(l<u)
{
j=partition(a,l,u);
quick_sort(a,l,j-1);
quick_sort(a,j+1,u);


}
}
int partition(int a[],int l,int u)
{
int v,i,j,temp;
v=a[l];
i=l;
j=u+1;
do
{
do
i++;
while(a[i]<v&&i<=u);
do
j--;
```

```
while(v<a[j]);

if(i<j)

{

temp=a[i];

a[i]=a[j];

a[j]=temp;

}

}while(i<j);

a[l]=a[j];

a[j]=v;

return(j);

}
```

output:

C:\Users\student\Desktop\exp3.exe

```
Ajay Dakhore [A24]
How many elements?7

Enter array elements:12 24 9 25 23 56 20

Array after sorting:9 12 20 23 24 25 56
Process returned 0 (0x0)   execution time : 37.535 s
Press any key to continue.
```

Experiment no.05

```
//Dijkstras algorithm

// AJAY DAKHORE[A24]

#include<iostream>

#include<conio.h>

#include<stdio.h>

using namespace std;

int shortest(int ,int);

int cost[10][10],dist[20],i,j,n,k,m,S[20],v,totcost,path[20],p;

main()

{

cout<<"AJAY DAKHORE[A24]";

int c;

cout <<"\nenter no of vertices";

cin >> n;

cout <<"enter no of edges";

cin >>m;

cout <<"\nenter\nEDGE Cost\n";

for(k=1;k<=m;k++)

{

cin >> i >> j >>c;

cost[i][j]=c;

}
```

```
for(i=1;i<=n;i++)

for(j=1;j<=n;j++)

if(cost[i][j]==0)

cost[i][j]=31999;

cout <<"enter initial vertex";

cin >>v;

cout << v<<"\n";

shortest(v,n);

}

int shortest(int v,int n)

{

int min;

for(i=1;i<=n;i++)

{

S[i]=0;

dist[i]=cost[v][i];

}

path[++p]=v;


S[v]=1;

dist[v]=0;

for(i=2;i<=n;i++)

{

k=-1;

min=31999;
```

```
for(j=1;j<=n;j++)
{
if(dist[j]<min && S[j]!=1)
{
min=dist[j];
k=j;
}
}
if(cost[v][k]<=dist[k])
p=1;
path[++p]=k;
for(j=1;j<=p;j++)
cout<<path[j];
cout <<"\n";
//cout <<k;
S[k]=1;

for(j=1;j<=n;j++)
if(cost[k][j]!=31999 && dist[j]>=dist[k]+cost[k][j] && S[j]!=1)
dist[j]=dist[k]+cost[k][j];
}
}
```

output:

C:\Users\Student\Desktop\ajay.exe

```
AJAY DAKHORE[A24]
enter no of vertices3
enter no of edges3

enter
EDGE Cost
1 2 56
2 3 10
1 3 78
enter initial vertex1
1
12
123

Process returned 0 (0x0)   execution time : 22.869 s
Press any key to continue.
```

Experiment no.06

```
//Program: Knapsack problem

//AJAY DAKHORE[A24]

#include <iostream>

using namespace std;

int knaps(int n,int m,int w[],int p[])

{

int i,j;

int knapsack[n+1][m+1];

for(j=0;j<=m;j++)

knapsack[0][j]=0;

for(i=0;i<=n;i++)

knapsack[i][0]=0;

for(i=1;i<=n;i++)

{

for(j=1;j<=m;j++)

{

if(w[i-1]<=j)

knapsack[i][j]=max(knapsack[i-1][j],p[i-1]+knapsack[i-1][j-w[i-1]]);

else

knapsack[i][j]=knapsack[i-1][j];

}

}
```

```
return knapsack[n][m];

}

int main()

{

int i,j,n,m;

cout<<"\nEnter number of item:\t";

cin>>n;

int w[n];

int p[n];

cout<<"\nEnetr weight & price of items:\t";

for(i=0;i<n;i++)

{

cin>>w[i]>>p[i];

}

cout<<"\nEnter capacity of knapsack:\t";

cin>>m;

int result=knaps(n,m,w,p);

cout<<"\nMaximum value that can be stored is:\t"<<result;

return 0;

}
```

output:

"C:\Users\Student\Desktop\Desktop files\Dheeraj practical oop\a.exe"

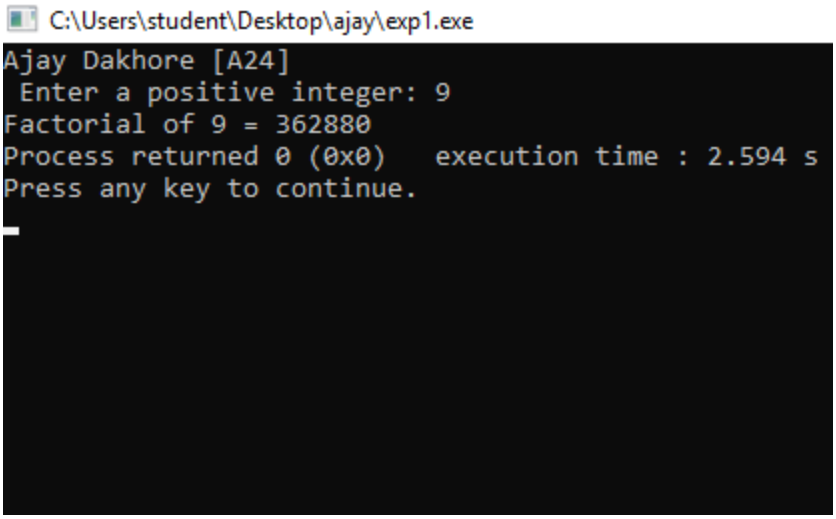
```
Enter number of item:   3

Enetr weight & price of items:
2 1
3 2
4 5

Enter capacity of knapsack:     6

Maximum value that can be stored is:    6
Process returned 0 (0x0)   execution time : 18.783 s
Press any key to continue.
```

Experiment no.01

```
// A24) AJAY DAKHORE

//1)Factorial using recursive function

#include<iostream>

using namespace std;

int factorial(int n);

int main()

{

int n;

cout<<"Ajay Dakhore [A24]";

cout << "\n Enter a positive integer: ";

cin >> n;

cout << "Factorial of " << n << " = " << factorial(n);

return 0;

}

int factorial(int n)

{

if(n > 1)

return n * factorial(n - 1);

else

return 1;
```

}

output:




```
// A24) AJAY DAKHORE

//2) factorial using iterative function

#include <iostream>

using namespace std;

int fact_iter(int n)

{

int result = 1;

for (int i = 1; i <= n; i++)

{

result *= i;

}

return result;

}

int main()

{
```

```
int n;

cout<<"Ajay Dakhore [A24]";

while (1)

{

cout<<"\n Enter interger (0 to exit): ";

cin>>n;

if (n == 0)

break;

cout<<fact_iter(n)<<endl;

}

return 0;

}
```

output:

C:\Users\student\Desktop\exp11.exe

```
Ajay Dakhore [A24]
 Enter interger (0 to exit): 9
362880

 Enter interger (0 to exit): 2
2

 Enter interger (0 to exit): 6
720

 Enter interger (0 to exit):
```

Experiment no.08

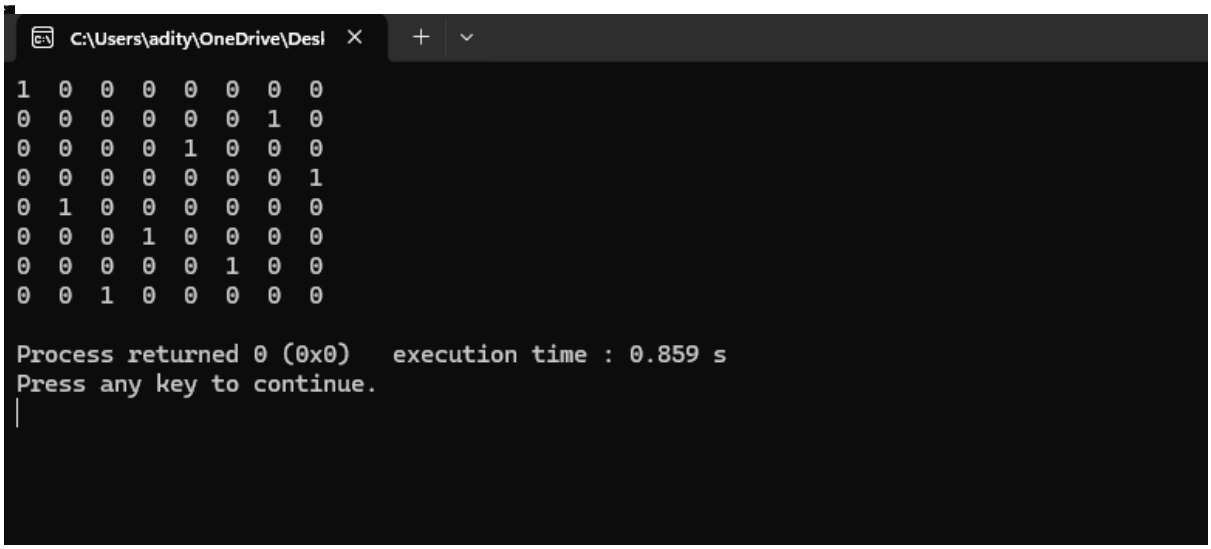
```
//AJAY DAKHORE[A24]

//C++ Program to Solve N-Queen Problem


#include <iostream>

#include <cstdio>

#include <cstdlib>

#define N 8

using namespace std;


/* print solution */

void printSolution(int board[N][N])

{

   for (int i = 0; i < N; i++)

   {

      for (int j = 0; j < N; j++)

         cout<<board[i][j]<<"  ";

      cout<<endl;

   }

}


/* check if a queen can be placed on board[row][col]*/

bool isSafe(int board[N][N], int row, int col)

{

   int i, j;
```

```
    for (i = 0; i < col; i++)
    {
        if (board[row][i])
            return false;
    }
    for (i = row, j = col; i >= 0 && j >= 0; i--, j--)
    {
        if (board[i][j])
            return false;
    }

    for (i = row, j = col; j >= 0 && i < N; i++, j--)
    {
        if (board[i][j])
            return false;
    }

    return true;
}

/*solve N Queen problem */
bool solveNQUtil(int board[N][N], int col)
{
    if (col >= N)
        return true;
    for (int i = 0; i < N; i++)
    {
```

```
        if ( isSafe(board, i, col) )
        {
            board[i][col] = 1;
            if (solveNQUtil(board, col + 1) == true)
                return true;
            board[i][col] = 0;
        }
    }
    return false;
}

/* solves the N Queen problem using Backtracking.*/
bool solveNQ()
{
    int board[N][N] = {0};
    if (solveNQUtil(board, 0) == false)
    {
        cout<<"Solution does not exist"<<endl;
        return false;
    }
    printSolution(board);
    return true;
}

// Main
int main()
{
```

```
    solveNQ();

    return 0;

}
```

OUTPUT:



Experiment no.04

```
// Prims algorithm.


#include <iostream>

using namespace std;

int i,j,k,a,b,v,u,n,ne=1;

int low,mincost=0,cost[10][10];

int visited[10]={0};

int main()

{

   cout<<"Prims algorithm\n";

   cout<<"\nEnter number of vertices:\t";

   cin>>n;

   cout<<"\nEnter the adjacency matrix:\n";

   for(i=1;i<=n;i++)

   {

      for(j=1;j<=n;j++)

      {

         cin>>cost[i][j];

         if(cost[i][j]==0)

            cost[i][j]=999;

      }

   }

   visited[1]=1;
```

```
// printf("\nThe edges of Minimum Cost spanning tree are:\t");
 while(ne<n)
 {
    for(i=1,low=999;i<=n;i++)
    {
       for(j=1;j<=n;j++)
       {
          if(cost[i][j]<low)
          {
             if(visited[i]!=0)
             {
             low=cost[i][j];
             a=u=i;
             b=v=j;
             }
          }
       }
    }
    if(visited[u]==0||visited[v]==0)
    {
       cout<<"\n edge cost= "<<low;
       mincost+=low;
       visited[b]=1;
    }
    cost[a][b]=cost[b][a]=999;
 }
 cout<<"\nMinimum Cost= "<<mincost;
```

```
    return 0;

}
```

OUTPUT:

```
C:\Users\adity\OneDrive\Desl

Prims algorithm

Enter number of vertices:       4

Enter the adjacency matrix:
0 1 3 4
1 0 2 999
3 2 0 5
4 999 5 0

 edge cost= 1
 edge cost= 2
 edge cost= 4
```

Content Beyond Syllabus

Experiment no.09

```
//AJAY DAKHORE[A24]

#include <iostream>

#include <climits>

using namespace std;

// Function to find the most efficient way to multiply

// a given sequence of matrices

int MatrixChainMultiplication(int dims[], int i, int j)

{

// base case: one matrix

if (j <= i + 1) {

return 0;

}

// stores the minimum number of scalar multiplications (i.e., cost)

// needed to compute matrix `M[i+1] ... M[j] = M[i...j]`

int min = INT_MAX;

// take the minimum over each possible position at which the

// sequence of matrices can be split

/*

(M[i+1]) × (M[i+2].................M[j])

(M[i+1]M[i+2]) × (M[i+3............M[j])

...

...

(M[i+1]M[i+2]............M[j-1]) × (M[j])
```

```
*/
for (int k = i + 1; k <= j - 1; k++)
{
// recur for `M[i+1]...M[k]` to get an `i × k` matrix
int cost = MatrixChainMultiplication(dims, i, k);
// recur for `M[k+1]...M[j]` to get an `k × j` matrix
cost += MatrixChainMultiplication(dims, k, j);
// cost to multiply two `i × k` and `k × j` matrix
cost += dims[i] * dims[k] * dims[j];
if (cost < min) {
min = cost;
}
}
// return the minimum cost to multiply `M[j+1]...M[j]`
return min;
}
// Matrix Chain Multiplication Problem
int main()
{
// Matrix `M[i]` has dimension `dims[i-1] × dims[i]` for `i = 1...n`
// input is `10 × 30` matrix, `30 × 5` matrix, `5 × 60` matrix

int dims[] = { 10, 30, 5, 60 };
int n = sizeof(dims) / sizeof(dims[0]);
cout << "The minimum cost is " << MatrixChainMultiplication (dims, 0, n - 1);
return 0;
}
```

output:

C:\Users\student\Desktop\aj.exe

```
The minimum cost is 4500
Process returned 0 (0x0)   execution time : 0.047 s
Press any key to continue.
```