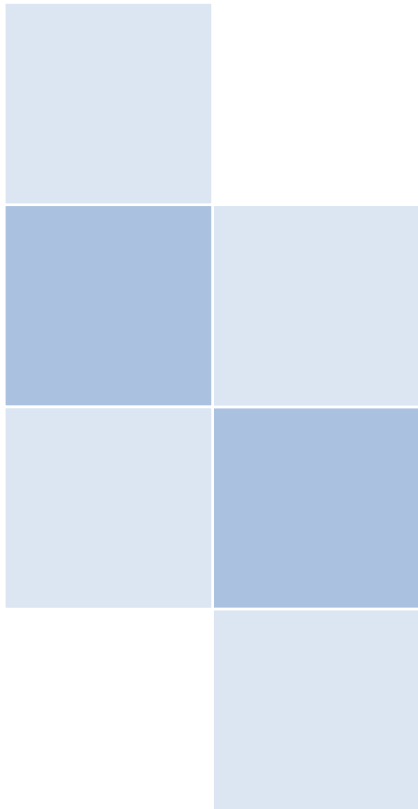


(06CSL58)

ALGORITHMS WITH C++ LAB MANUAL



Akash Devaraju

[DEPARTMENT OF CS/IS]

2011

Treating Code As An Essay

PROGRAMS SHARE SOME ATTRIBUTES WITH ESSAYS. For essays, the most important question readers ask is, “What is it about?” For programs, the main question is, “What does it do?” In fact, the purpose should be sufficiently clear that neither question ever needs to be uttered. Still, for both essays and computer code, it’s always important to look at how each one is written. Even if the idea itself is good, it will be difficult to transmit to the desired audience if it is difficult to understand. The style in which they are written is just as important as their purpose. Both essays and lines of code are meant—before all else—to be read and understood by human beings.

Computers can, of course, deal with complexity without complaint, but this is not the case for human beings. Unreadable code will reduce most people’s productivity significantly. On the other hand, easily understandable code will increase it. And we see beauty in such code.

“Beautiful Code” is not an abstract virtue that exists independent of its programmer’s efforts. Rather, beautiful code is really meant to help the programmer be happy & productive.

In other words, what is beautiful code?

Brevity: “Succinctness is power”, there’s a definite cost involved in scanning code with the human eye, and programs should ideally contain no unnecessary information. Eliminate Redundancy ...

DRY Principle: Don’t Repeat Yourself. If the same code exists in multiple places whatever you are trying to say becomes obscured ...

Simplicity: Feel beauty in simple code. If a program is hard to understand, it can’t be beautiful. And when programs are obscure rather than comprehensible, the results are bugs, mistakes & confusion.

Flexibility: “Freedom from enforcement of tools”. Humans are more valuable than any tools or languages, computers should serve programmers to maximise their productivity & happiness but in reality they often increase the burden instead of lightening it.

Balance is the final element of beautiful code. None of the above elements will by itself ensure a beautiful program. When balanced together & kept in mind from the very beginning, each element will work harmoniously with the others to create a beautiful code. And if you make sure to have fun writing and reading code, you will experience happiness as a programmer

Yukihiro (‘Matz’) Matsumoto

(Creator of the “Ruby” Language)

(Adapted from the book – “The Beautiful Code”)

Subject Code	:	06CSL58	IA Marks	:	25
No. of Practical Hrs./ Week	:	03	Exam Hours	:	03
Total No. of Practical Hrs.	:	42	Exam Marks	:	50

IMPLEMENT THE FOLLOWING USING C/C++ LANGUAGE:

1. Implement Recursive Binary search and Linear search and determine the time required to search an element. Repeat the experiment for different values of n, the number of elements in the list to be searched and plot a graph of the time taken versus n.
2. Sort a given set of elements using the Heapsort method and determine the time required to sort the elements. Repeat the experiment for different values of n, the number of elements in the list to be sorted and plot a graph of the time taken versus n.
3. Sort a given set of elements using Merge sort method and determine the time required to sort the elements. Repeat the experiment for different values of n, the number of elements in the list to be sorted and plot a graph of the time taken versus n.
4. Sort a given set of elements using Selection sort and determine the time required to sort elements. Repeat the experiment for different values of n, the number of elements in the list to be sorted and plot a graph of the time taken versus n.
5. Implement 0/1 Knapsack problem using dynamic programming.
6. From a given vertex in a weighted connected graph, find shortest paths to other vertices using Dijkstra's algorithm.
7. Sort a given set of elements using Quick sort method and determine the time required sort the elements. Repeat the experiment for different values of n, the number of elements in the list to be sorted and plot a graph of the time taken versus n.
8. Find Minimum Cost Spanning Tree of a given undirected graph using Kruskal's algorithm.
9.
 - a. Print all the nodes reachable from a given starting node in a digraph using BFS method.
 - b. Check whether a given graph is connected or not using DFS method.
10. Find a subset of a given set $S = \{s_1, s_2, \dots, s_n\}$ of n positive integers whose sum is equal to a given positive integer d. For example, if $S = \{1, 2, 5, 6, 8\}$ and $d = 9$ there are two solutions $\{1, 2, 6\}$ and $\{1, 8\}$. A suitable message is to be displayed if the given problem

instance doesn't have a solution.

11. a. Implement Horspool algorithm for String Matching.
b. Find the Binomial Co-efficient using Dynamic Programming.
12. Find Minimum Cost Spanning Tree of a given undirected graph using Prim's algorithm.
13. a. Implement Floyd's algorithm for the All-Pairs- Shortest-Paths problem.
b. Compute the transitive closure of a given directed graph using Warshall's algorithm.
14. Implement N Queen's problem using Back Tracking.

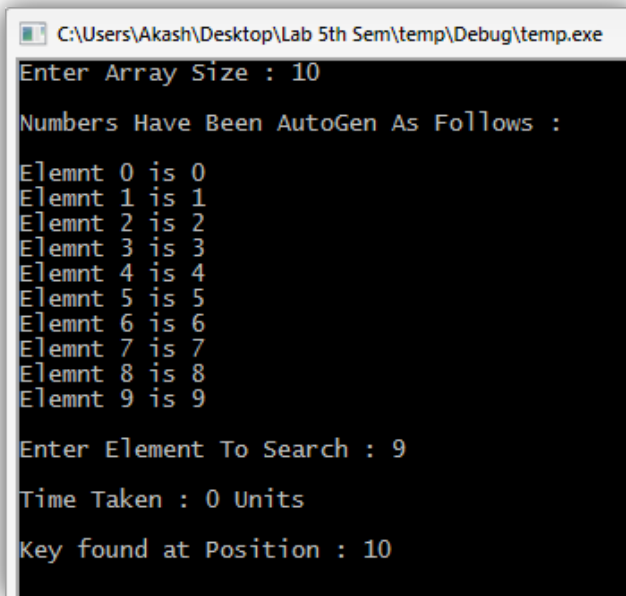
Contents

TREATING CODE AS AN ESSAY	1
LAB PROGRAM 1A	3
OUTPUT	4
LAB PROGRAM 1B	5
OUTPUT	6
LAB PROGRAM 2	7
OUTPUT	9
LAB PROGRAM 3	10
OUTPUT	12
LAB PROGRAM 4	13
OUTPUT	14
LAB PROGRAM 5	15
OUTPUT	16
LAB PROGRAM 6	17
OUTPUT	19
LAB PROGRAM 7	20
OUTPUT	22
LAB PROGRAM 8	23
OUTPUT	25
LAB PROGRAM 9A	26
OUTPUT	28
LAB PROGRAM 9B	29
OUTPUT	31
LAB PROGRAM 10	32
OUTPUT	34
LAB PROGRAM 11A	35
OUTPUT	37
LAB PROGRAM 11B	38
OUTPUT	39
LAB PROGRAM 12	40
OUTPUT	42
LAB PROGRAM 13A	43
OUTPUT	45
LAB PROGRAM 13B	46
OUTPUT	48
LAB PROGRAM 14	49
OUTPUT	51

Lab Program 1a

```
1  /**
2   Lab Program 1a
3   Recursive Binary Search
4   */
5
6   #include <iostream>
7   #include <conio.h>
8   #include <time.h>
9
10  #define MAX 10
11
12  using namespace std;
13
14  int binary(int key,int a[],int low,int high)
15  {
16      if(low>high)
17          return -1;
18      int mid=(low+high)/2;
19      if(key<a[mid])
20          return binary(key,a,low,mid-1);
21      else if(key>a[mid])
22          return binary(key,a,mid+1,high);
23      return mid;
24  }
25
26
27  void main()
28  {
29      int n,a[MAX],i,key;
30      time_t end,start;
31
32      cout<<"Enter Array Size : ";
33      cin>>n;
34
35      cout<<"\nNumbers Have Been AutoGen As Follows : \n";
36      for (i=0;i<n;i++)
37      {
38          a[i]=i;
39          cout<<"\nElemnt "<<i<<"is "<<a[i];
40      }
41
42      cout<<"\n\nEnter Element To Search : ";
43      cin>>key;
44
45      int pos;
46
47      time(&start);
48      pos=binary(key,a,0,n-1);
49      time(&end);
50
51      cout<<"\nTime Taken : "<<difftime(end,start)<<" Units";
52
53      if(pos==-1)
54          cout<<"\n\nUnsuccessful search\n";
55      else
56          cout<<"\n\nKey found at Position : "<<pos+1;
57
58      getch();
59  }
```

Output

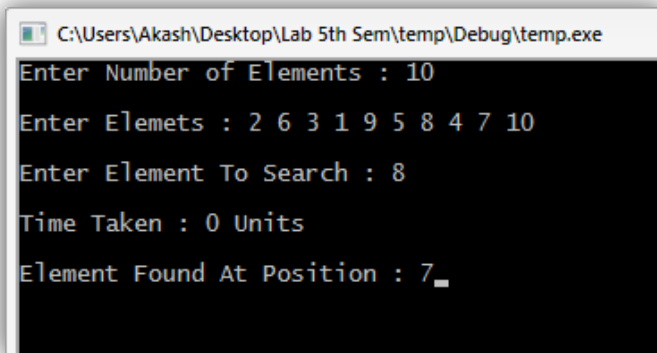


```
C:\Users\Akash\Desktop\Lab 5th Sem\temp\Debug\temp.exe
Enter Array Size : 10
Numbers Have Been AutoGen As Follows :
Elemnt 0 is 0
Elemnt 1 is 1
Elemnt 2 is 2
Elemnt 3 is 3
Elemnt 4 is 4
Elemnt 5 is 5
Elemnt 6 is 6
Elemnt 7 is 7
Elemnt 8 is 8
Elemnt 9 is 9
Enter Element To Search : 9
Time Taken : 0 Units
Key found at Position : 10
```

Lab Program 1b

```
1  /**
2   Lab Program 1b
3   Recursive Linear Search
4   Author SkyKOG
5   */
6
7   #include <iostream>
8   #include <conio.h>
9   #include <time.h>
10
11  #define MAX 10
12
13  using namespace std;
14
15  int linear(int a[],int key,int n)
16  {
17      if(n<0)
18          return -1;
19      if(key==a[n])
20          return n;
21      return linear(a,key,n-1);
22  }
23
24  void main()
25  {
26      int a[MAX],n,key,pos;
27      time_t start,end;
28
29      cout<<"Enter Number of Elements : ";
30      cin>>n;
31
32      cout<<"\nEnter Elemets : ";
33      for(int i=0;i<n;i++)
34          cin>>a[i];
35
36      cout<<"\nEnter Element To Search : ";
37      cin>>key;
38
39      time(&start);
40      pos=linear(a,key,n);
41      time(&end);
42
43      cout<<"\nTime Taken : "<<difftime(end,start)<<" Units";
44
45      if(pos== -1)
46          cout<<"\n\nNot Found";
47      else
48          cout<<"\n\nElement Found At Position : "<<pos+1;|
49
50      getch();
51  }
```


Output



A screenshot of a Windows command prompt window. The title bar shows the file path: C:\Users\Akash\Desktop\Lab 5th Sem\temp\Debug\temp.exe. The window has a black background with white text. The text inside the window shows the following sequence of prompts and user input:

```
Enter Number of Elements : 10
Enter Elements : 2 6 3 1 9 5 8 4 7 10
Enter Element To Search : 8
Time Taken : 0 Units
Element Found At Position : 7_
```

Lab Program 2

```
1  /**
2   Lab Program 2
3   Heapsort
4   Author SkyKOG
5   */
6
7   #include <iostream>
8   #include <conio.h>
9   #include <time.h>
10
11  #define MAX 10
12
13  using namespace std;
14
15  void heapsort(int [], int );
16
17  void main()
18  {
19      int i,n,a[MAX];
20      clock_t start,end;
21
22      a[0]=0;
23
24      cout<<"Enter Num of Elements : ";
25      cin>>n;
26
27      cout<<"\nEnter Elements : ";
28      for(i=1;i<=n;i++)
29          cin>>a[i];
30
31      start=clock();
32      heapsort(a, n);
33      end=clock();
34
35      cout<<"\nThe Sorted Elements Are : ";
36      for(i = 1; i < n+1; i++)
37          cout<<a[i]<<" ";
38
39      cout<<"\n\nTime Taken : "<<(end-start)/(double)CLOCKS_PER_SEC;
40      getch();
41  }
42
```

```

43 void heapsort(int H[], int n)
44 {
45     int i,k,v,j;
46
47     for(i = n/2; i >= 1; i--)
48     {
49         k = i; v = H[k];
50         bool heap = false;
51         while(!heap && 2*k<=n)
52         {
53             j = 2*k;
54             if(j < n)
55                 if(H[j] < H[j+1])
56                     j = j+1;
57             if(v >= H[j])
58                 heap = true;
59             else
60             {
61                 H[k] = H[j];
62                 k = j;
63             }
64         }
65         H[k] = v;
66     }
67     for(int i=n-1;i>0;i--)
68     {
69         int temp=H[n];
70         H[n]=H[1];
71         H[1]=temp;
72         n--;
73         heapsort(H,n);
74     }
75     return;
76 }

```

Output

```
C:\Users\Akash\Desktop\Lab 5th Sem\temp\Debug\temp.exe
Enter Num of Elements : 5
Enter Elements : 5 9 6 2 1
The Sorted Elements Are : 1 2 5 6 9
Time Taken : 0_
```

```
C:\Users\Akash\Desktop\Lab 4th Sem\temp\Debug\temp.exe
Enter Num of Elements : 1000
Enter Elements :
The Sorted Elements Are :
Time Taken : 0_
```

```
C:\Users\Akash\Desktop\Lab 4th Sem\temp\Debug\temp.exe
Enter Num of Elements : 4000
Enter Elements :
The Sorted Elements Are :
Time Taken : 0.015_
```

```
C:\Users\Akash\Desktop\Lab 4th Sem\temp\Debug\temp.exe
Enter Num of Elements : 50000
Enter Elements :
The Sorted Elements Are :
Time Taken : 0.016
```

```
C:\Users\Akash\Desktop\Lab 4th Sem\temp\Debug\temp.exe
Enter Num of Elements : 99999
Enter Elements :
The Sorted Elements Are :
Time Taken : 0.046
```

Lab Program 3

```
1  /**
2   Lab Program 3
3   Merge Sort
4   Author SkyKOG
5   */
6
7   #include <iostream>
8   #include <conio.h>
9   #include <time.h>
10
11  #define MAX 10
12
13  using namespace std;
14
15  void msort(int a[],int low,int high);
16  void merge(int a[],int low,int mid,int high);
17
18  void main()
19  {
20      int i,n,a[MAX];
21      clock_t start,end;
22
23      cout<<"Enter Num of Elements : ";
24      cin>>n;
25
26      cout<<"\nEnter Elements : ";
27      for(i = 0 ; i < n ; i++)
28          cin>>a[i];
29
30      start=clock();
31      msort(a,0,n-1);
32      end=clock();
33
34      cout<<"\nThe Sorted Elements Are : ";
35      for(i = 0 ; i < n ; i++)
36          cout<<a[i]<<" ";
37
38      cout<<"\n\nTime Taken : "<<(end-start)/(double)CLOCKS_PER_SEC;
39      getch();
40  }
```

```

42 void merge(int a[],int low,int mid,int high)
43 {
44     int i=low;
45     int j=mid+1;
46     int k=low;
47     int temp[MAX];
48
49     while(i<=mid && j<=high)
50     {
51         if(a[i]<a[j])
52         {
53             temp[k]=a[i];
54             i++;k++;
55         }
56         else
57         {
58             temp[k]=a[j];
59             j++;k++;
60         }
61     }
62
63     while(i<=mid)
64         temp[k++]=a[i++];
65
66     while(j<=high)
67         temp[k++]=a[j++];
68
69     for(i=low;i<=high;i++)
70         a[i]=temp[i];
71 }
72
73 void msort(int a[],int low,int high)
74 {
75     int mid;
76
77     if(low<high)
78     {
79         mid=(low+high)/2;
80         msort(a,low,mid);
81         msort(a,mid+1,high);
82         merge(a,low,mid,high);
83     }
84 }

```

Output

```
C:\Users\Akash\Desktop\Lab 5th Sem\temp\Debug\temp.exe
Enter Num of Elements : 6
Enter Elements : 9 5 4 7 6 3
The Sorted Elements Are : 3 4 5 6 7 9
Time Taken : 0
```

```
C:\Users\Akash\Desktop\Lab 4th Sem\Prg 3\Debug\Prg
Enter Num of Elements : 1000
Enter Elements :
The Sorted Elements Are :
Time Taken : 0.031_
```

```
C:\Users\Akash\Desktop\Lab 4th Sem\Prg 3\Debug\Prg 3.exe
Enter Num of Elements : 4000
Enter Elements :
The Sorted Elements Are :
Time Taken : 0.11_
```

```
C:\Users\Akash\Desktop\Lab 4th Sem\Prg 3\Debug\Prg 3.exe
Enter Num of Elements : 50000
Enter Elements :
The Sorted Elements Are :
Time Taken : 1.186_
```

```
C:\Users\Akash\Desktop\Lab 4th Sem\Prg 3\Debug\Prg 3.exe
Enter Num of Elements : 99999
Enter Elements :
The Sorted Elements Are :
Time Taken : 2.465_
```

Lab Program 4

```
1  /**
2   Lab Program 4
3   Selection Sort
4   Author SkyKOG
5   */
6
7   #include <iostream>
8   #include <conio.h>
9   #include <time.h>
10
11  #define MAX 10
12  |
13  using namespace std;
14
15  void selsort(int [],int);
16
17  void main()
18  {
19      int i,n,a[MAX];
20      clock_t start,end;
21
22      cout<<"Enter Num of Elements : ";
23      cin>>n;
24
25      cout<<"\nEnter Elements : ";
26      for(i = 0 ; i < n ; i++)
27          cin>>a[i];
28
29      start=clock();
30      selsort(a, n);
31      end=clock();
32
33      cout<<"\nThe Sorted Elements Are : ";
34      for(i = 0 ; i < n ; i++)
35          cout<<a[i]<<" ";
36
37      cout<<"\n\nTime Taken : "<<(end-start)/(double)CLOCKS_PER_SEC;
38      getch();
39  }
40
41  void selsort(int a[],int n)
42  {
43      for(int i = 0 ; i < n-1 ; i++)
44      {
45          for(int j = i+1 ; j < n ; j++)
46          {
47              if(a[i] > a[j])
48              {
49                  int temp = a[i];
50                  a[i] = a[j];
51                  a[j] = temp;
52              }
53          }
54      }
55  }
```


Output

```
C:\Users\Akash\Desktop\Lab 5th Sem\temp\Debug\temp.exe
Enter Num of Elements : 8
Enter Elements : 9 5 1 2 6 3 8 4
The Sorted Elements Are : 1 2 3 4 5 6 8 9
Time Taken : 0_
```

```
C:\Users\Akash\Desktop\Lab 4th Sem\Prg 4\Debug\Prg 4.exe
Enter Num of Elements : 1000
Enter Elements :
Time Taken : 0.016
```

```
C:\Users\Akash\Desktop\Lab 4th Sem\Prg 4\Debug\Prg 4.exe
Enter Num of Elements : 4000
Enter Elements :
Time Taken : 0.109_
```

```
C:\Users\Akash\Desktop\Lab 4th Sem\Prg 4\Debug\Prg 4.exe
Enter Num of Elements : 50000
Enter Elements :
Time Taken : 13.088
```

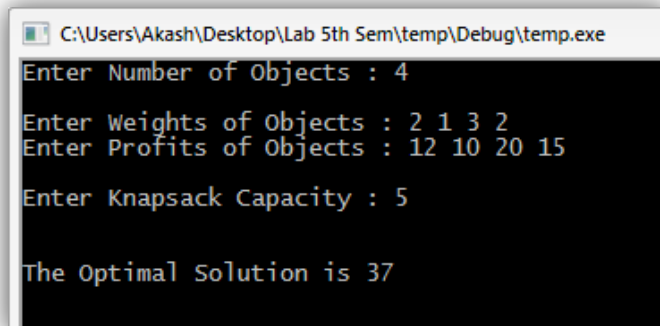
```
C:\Users\Akash\Desktop\Lab 4th Sem\Prg 4\Debug\Prg 4.exe
Enter Num of Elements : 99999
Enter Elements :
Time Taken : 45.427_
```

Lab Program 5

```
1  /**
2   Lab Program 5
3   1/0 Knapsack - Dynamic Programming
4   Author SkyKOG
5   */
6
7   #include <iostream>
8   #include <conio.h>
9
10  using namespace std;
11
12  int w[10],p[10],n,m;
13
14  int knap(int i,int j);
15  int max(int a,int b);
16
17  void main()
18  {
19      int i;
20
21      cout<<"Enter Number of Objects : ";
22      cin>>n;
23
24      cout<<"\nEnter Weights of Objects : ";
25      for(i=1;i<=n;i++)
26          cin>>w[i];
27
28      cout<<"Enter Profits of Objects : ";
29      for(i=1;i<=n;i++)
30          cin>>p[i];
31
32      cout<<"\nEnter Knapsack Capacity : ";
33      cin>>m;
34
35      int max=knap(n,m);
36
37      cout<<"\n\nThe Optimal Solution is "<<max;
38
39      getch();
40  }
41
42  int max(int a,int b)
43  {
44      return a>b?a:b;
45  }
46
47  int knap(int i,int j)
48  {
49      if(i==0||j==0)
50          return 0;
51      if(w[i]>j)
52          return knap(i-1,j);
53      return
54          max(knap(i-1,j),knap(i-1,j-w[i])+p[i]);
55  }
```

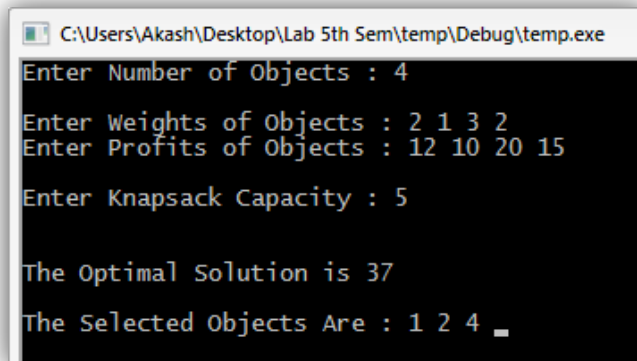
Output

Dynamic Programming – Memory Functions



```
C:\Users\Akash\Desktop\Lab 5th Sem\temp\Debug\temp.exe
Enter Number of Objects : 4
Enter Weights of Objects : 2 1 3 2
Enter Profits of Objects : 12 10 20 15
Enter Knapsack Capacity : 5
The Optimal Solution is 37
```

Simple 1/0 Knapsack



```
C:\Users\Akash\Desktop\Lab 5th Sem\temp\Debug\temp.exe
Enter Number of Objects : 4
Enter Weights of Objects : 2 1 3 2
Enter Profits of Objects : 12 10 20 15
Enter Knapsack Capacity : 5
The Optimal Solution is 37
The Selected Objects Are : 1 2 4 _
```

Lab Program 6

```
1  /**
2   Lab Program 6
3   Dijkstra Algorithm - Greedy
4   Author SkyKOG
5   */
6
7   #include <iostream>
8   #include <conio.h>
9
10  #define MAX 50
11  #define INFINITY 999
12
13  using namespace std;
14
15  int n,dist[MAX][MAX],d[MAX];
16
17  void display();
18  void dijkstra(int);
19
20  void main()
21  {
22      int i, j;
23
24      cout<<"Enter Number of Edges : ";
25      cin>>n;
26
27      cout<<"\nEnter Cost Matrix 0-No Edge : \n\n";
28      for (i = 1; i <=n; ++i)
29          for (j = 1; j <=n; ++j)
30              cin>>dist[i][j];
31
32      i=1;
33      while(i<=n)
34      {
35          cout<<"\nSource is : "<<i<<"\n";
36          dijkstra(i);
37          display();
38          getch();i++;
39      }
40  }
41
42  void display()
43  {
44      cout<<"\n";
45      for (int i = 1; i <= n; ++i)
46          cout<<"\t"<<i;
47      cout<<"\n";
48      for (int i = 1; i <= n; ++i)
49          cout<<"\t"<<d[i];
50      cout<<"\n";
51  }
```

```

53 void dijkstra(int s)
54 {
55     int i, k, mini;
56     int visited[MAX];
57
58     for (i = 1; i <= n; ++i)
59     {
60         d[i] = INFINITY;
61         visited[i] = 0;
62     }
63
64     d[s] = 0;
65
66     for (k = 1; k <= n; ++k)
67     {
68         mini = -1;
69         for (i = 1; i <= n; ++i)
70             if (!visited[i] && ((mini == -1) || (d[i] < d[mini])))
71                 mini = i;
72
73         visited[mini] = 1;
74
75         for (i = 1; i <= n; ++i)
76             if (dist[mini][i])
77                 if (d[mini] + dist[mini][i] < d[i])
78                     d[i] = d[mini] + dist[mini][i];
79     }
80 }

```

Output

```
C:\Users\Akash\Desktop\Lab 5th Sem\temp\Debug\temp.exe
Enter Number of Edges : 6
Enter Cost Matrix 0-No Edge :
0 15 10 0 45 0
0 0 15 0 20 0
20 0 0 20 0 0
0 10 0 0 35 0
0 0 0 30 0 0
0 0 0 4 0 0

Source is : 1
    1    2    3    4    5    6
    0    15   10   30   35  999

Source is : 2
    1    2    3    4    5    6
   35    0   15   35   20  999

Source is : 3
    1    2    3    4    5    6
   20   30    0   20   50  999

Source is : 4
    1    2    3    4    5    6
   45   10   25    0   30  999

Source is : 5
    1    2    3    4    5    6
   75   40   55   30    0  999

Source is : 6
    1    2    3    4    5    6
   49   14   29    4   34    0
```

Lab Program 7

```
1  /**
2   Lab Program 7
3   Quick Sort
4   Author SkyKOG
5   */
6
7   #include <iostream>
8   #include <conio.h>
9   #include <time.h>
10
11  #define MAX 100000
12
13  using namespace std;
14
15  int split ( int a[ ], int lower, int upper );
16  void quicksort ( int a[ ], int lower, int upper );
17
18  void main( )
19  {
20      int i,n,a[MAX];
21      clock_t start,end;
22
23      cout<<"Enter Num of Elements : ";
24      cin>>n;
25
26      cout<<"\nEnter Elements : ";
27      for(i = 0 ; i < n ; i++)
28          cin>>a[i];
29
30      start=clock();
31      quicksort(a,0,n-1);
32      end=clock();
33
34      cout<<"\nThe Sorted Elements Are : ";
35      for(i = 0 ; i < n ; i++)
36          cout<<a[i]<<" ";
37
38      cout<<"\n\nTime Taken : "<<(end-start)/(double)CLOCKS_PER_SEC;
39      getch();
40  }
41
42  void quicksort ( int a[ ], int lower, int upper )
43  {
44      int i ;
45      if ( upper > lower )
46      {
47          i = split ( a, lower, upper ) ;
48          quicksort ( a, lower, i - 1 ) ;
49          quicksort ( a, i + 1, upper ) ;
50      }
51  }
```

```

53 int split ( int a[ ], int lower, int upper )
54 {
55     int p,q,i,t;
56
57     p = lower + 1 ;
58     q = upper ;
59     i = a[lower] ;
60
61     while ( q >= p )
62     {
63         while ( a[p] <= i )
64             p++ ;
65
66         while ( a[q] > i )
67             q-- ;
68
69         if ( q > p )
70         {
71             t = a[p] ;
72             a[p] = a[q] ;
73             a[q] = t ;
74         }
75     }
76
77     t = a[lower] ;
78     a[lower] = a[q] ;
79     a[q] = t ;
80
81     return q ;
82 }

```


Output

```
C:\Users\Akash\Desktop\Lab 5th Sem\temp\Debug\temp.exe
Enter Num of Elements : 5
Enter Elements : 1 5 9 8 4
The Sorted Elements Are : 1 4 5 8 9
Time Taken : 0_
```

```
C:\Users\Akash\Desktop\Lab 4th Sem\Prg 7\Debug\Prg 7.exe
Enter Num of Elements : 1000
Enter Elements :
Time Taken : 0.001
```

```
C:\Users\Akash\Desktop\Lab 4th Sem\Prg 7\Debug\Prg 7.exe
Enter Num of Elements : 4000
Enter Elements :
Time Taken : 0.002
```

```
C:\Users\Akash\Desktop\Lab 4th Sem\Prg 7\Debug\Prg 7.exe
Enter Num of Elements : 50000
Enter Elements :
Time Taken : 0.029
```

```
C:\Users\Akash\Desktop\Lab 4th Sem\Prg 7\Debug\Prg 7.exe
Enter Num of Elements : 99999
Enter Elements :
Time Taken : 0.041
```

Lab Program 8

```
1  /**
2   Lab Program 8
3   Kruskal's Algorithm - Greedy
4   Author SkyKOG
5   */
6
7   #include <iostream>
8   #include <conio.h>
9
10  #define MAX 20
11
12  using namespace std;
13
14  int parent[MAX],cost[MAX][MAX],t[MAX][2];
15
16  void kruskal(int);
17  int find(int);
18  void unite(int,int);
19
20  void main()
21  {
22      int i,j,n;
23
24      cout<<"Enter Number of vertices : ";
25      cin>>n;
26
27      for(i=1;i<=n;i++)
28          parent[i]=0;
29
30      cout<<"\nEnter Cost Matrix : \n\n";
31      for(i=1;i<=n;i++)
32      {
33          for(j=1;j<=n;j++)
34              cin>>cost[i][j];
35      }
36
37      kruskal(n);
38
39      getch();
40  }
41
42  int find(int v)
43  {
44      while(parent[v])
45          v=parent[v];
46      return v;
47  }
48
49  void unite(int i,int j)
50  {
51      parent[j]=i;
52  }
```

```

54 void kruskal(int n)
55 {
56     int i,j,k,u,v,mincost,res1,res2,sum=0;
57
58     for(k=1;k<n;k++)
59     {
60         mincost=999;
61         for(i=1;i<n-1;i++)
62         {
63             for(j=1;j<=n;j++)
64             {
65                 if(i==j)continue;
66                 if(cost[i][j]<mincost)
67                 {
68                     u=find(i);
69                     v=find(j);
70                     if(u!=v)
71                     {
72                         res1=i;
73                         res2=j;
74                         mincost=cost[i][j];
75                     }
76                 }
77             }
78         }
79         unite(res1,find(res2));
80         t[k][1]=res1;
81         t[k][2]=res2;
82         sum=sum+mincost;
83     }
84     cout<<"\nCost : "<<sum;
85     cout<<"\n\nEdges : \n\n";
86     for(i=1;i<n;i++)
87         cout<<t[i][1]<<" -> "<<t[i][2]<<"\n";
88 }

```

Output

```
C:\Users\Akash\Desktop\Lab 5th Sem\temp\Debug\temp.exe
Enter Number of vertices : 4
Enter Cost Matrix 0-No Edge 999-No Reach :
0 20 2 999
20 0 15 5
2 15 0 25
999 5 25 0

Cost : 22

Edges :
1 -> 3
2 -> 4
2 -> 3
```

Lab Program 9a

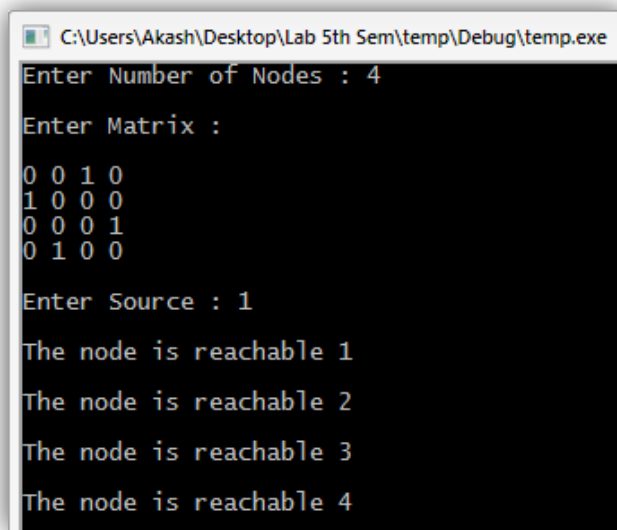
```
1  /**
2   Lab Program 9a
3   BFS Traversal - Decrease And Conquer
4   Author SkyKOG
5   */
6
7   #include <iostream>
8   #include <conio.h>
9
10  #define MAX 10
11
12  using namespace std;
13
14  class BFS {
15
16      int visited[MAX],n,a[MAX][MAX],i,j,source;
17
18      public:
19          void input();
20          void bfstraverse();
21          void display();
22  };
23
24  void main()
25  {
26      BFS obj;
27
28      obj.input();
29      obj.bfstraverse();
30      obj.display();
31
32      getch();
33  }
34
35  void BFS::display()
36  {
37      for(i=1;i<=n;i++)
38      {
39          if(visited[i])
40              cout<<"\nThe node is reachable "<<i<<"\n";
41          else
42              cout<<"\nNot Reachable "<<i<<"\n";
43      }
44  }
```

```

46 void BFS::input()
47 {
48     cout<<"Enter Number of Nodes : ";
49     cin>>n;
50
51     cout<<"\nEnter Matrix : \n\n";
52     for(i=1;i<=n;i++)
53         for(j=1;j<=n;j++)
54             cin>>a[i][j];
55
56     cout<<"\nEnter Source : ";
57     cin>>source;
58
59     for(i=1;i<=n;i++)
60         visited[i]=0;
61 }
62 void BFS::bfstraverse()
63 {
64     int i,queue[MAX],u,front=1,rear=1;
65
66     queue[rear]=source;
67
68     while(front<=rear)
69     {
70         u=queue[front++];
71         for(i=1;i<=n;i++)
72         {
73             if(a[u][i] && !visited[i])
74             {
75                 queue[++rear]=i;
76                 visited[i]=1;
77             }
78         }
79     }
80 }

```

Output



```
C:\Users\Akash\Desktop\Lab 5th Sem\temp\Debug\temp.exe
Enter Number of Nodes : 4
Enter Matrix :
0 0 1 0
1 0 0 0
0 0 0 1
0 1 0 0
Enter Source : 1
The node is reachable 1
The node is reachable 2
The node is reachable 3
The node is reachable 4
```

Lab Program 9b

```
1  /**
2   Lab Program 9b
3   DFS Connected - Decrease And Conquer
4   Author SkyKOG
5   */
6
7   #include <iostream>
8   #include <conio.h>
9
10  #define MAX 10
11
12  using namespace std;
13
14  class DFS {
15
16      int i,j,source,visited[MAX],n,a[MAX][MAX],count;
17
18      public:
19          void input();
20          void dfsconn(int,int a[MAX][MAX],int);
21          void diplay();
22  };
23
24  void main()
25  {
26      DFS obj;
27
28      obj.input();
29      obj.diplay();
30
31      getch();
32  }
33
34  void DFS::input()
35  {
36      cout<<"Enter Number of Nodes : ";
37      cin>>n;
38
39      cout<<"\nEnter Matrix : \n\n";
40      for(i=1;i<=n;i++)
41          for(j=1;j<=n;j++)
42              cin>>a[i][j];
43
44      cout<<"\nEnter Source : ";
45      cin>>source;
46
47      for(i=1;i<=n;i++)
48          visited[i]=0;
49
50      dfsconn(n,a,source);
51  }
```



```

53 void DFS::dfsconn(int n,int a[MAX][MAX],int source)
54 {
55     visited[source]=1;
56
57     for(i=1;i<=n;i++)
58         if(a[source][i] && !visited[i])
59             dfsconn(n,a,i);
60 }
61
62 void DFS::diplay()
63 {
64     count=0;
65
66     for(i=1;i<=n;i++)
67         if(visited[i])
68             count++;
69
70     if(count==n)
71         cout<<"\nConnected Graph\n";
72     else
73         cout<<"\nDisconnected Graph\n";
74 }

```

Output

```
C:\Users\Akash\Desktop\Lab 5th Sem\temp\Debu
Enter Number of Nodes : 4
Enter Matrix :
0 1 1 0
1 0 0 0
1 0 0 1
0 0 1 0

Enter Source : 2
Connected Graph
```

```
C:\Users\Akash\Desktop\Lab 5th Sem\temp\Debu
Enter Number of Nodes : 4
Enter Matrix :
0 1 1 0
1 0 0 0
1 0 0 0
0 0 0 0

Enter Source : 1
Disconnected Graph
```

Lab Program 10

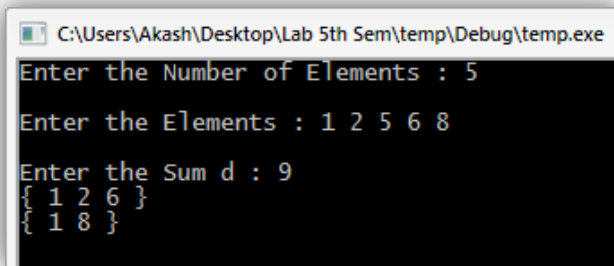
```
1  /**
2   Lab Program 10
3   Generating Subsets - Decrease And Conquer
4   */
5
6   #include <iostream>
7   #include <conio.h>
8   #include <cmath>
9
10  using namespace std;
11
12  void input();
13  void subset();
14  void display();
15
16  int set[100], sub[100], sum, i, n, count, flag = 0;
17
18  void main()
19  {
20      input();
21
22      subset();
23
24      if(!flag)
25          cout<<"\nNo solutions found.";
26
27      getch();
28  }
29
30  void input()
31  {
32      cout<<"Enter the Number of Elements : ";
33      cin>>n;
34
35      cout<<"\nEnter the Elements : ";
36      for(i = 0; i < n; i++)
37          cin>>set[i];
38
39      cout<<"\nEnter the Sum d : ";
40      cin>>sum;
41  }
```

```

43 void subset()
44 {
45     int mask = pow(2.0,n);
46     int total, j;
47
48     for(i = 0; i < mask; i++)
49     {
50         count = total = 0;
51
52         for(j = 0; j <= i; j++)
53         {
54             if(i & (1 << j))
55             {
56                 total += set[j];
57                 sub[count] = set[j];
58                 count++;
59             }
60         }
61         if(total == sum)
62             display();
63     }
64 }
65
66 void display()
67 {
68     flag = 1;
69     cout<<"{";
70     for(int j = 0; j < count; j++)
71         cout<<" "<<sub[j];
72     cout<<" }\n";
73 }

```

Output



A screenshot of a Windows command prompt window. The title bar shows the file path: C:\Users\Akash\Desktop\Lab 5th Sem\temp\Debug\temp.exe. The command prompt has a black background with white text. It displays the following text: 'Enter the Number of Elements : 5', 'Enter the Elements : 1 2 5 6 8', 'Enter the Sum d : 9', and two lines of array representation: '{ 1 2 6 }' and '{ 1 8 }'.

```
C:\Users\Akash\Desktop\Lab 5th Sem\temp\Debug\temp.exe
Enter the Number of Elements : 5
Enter the Elements : 1 2 5 6 8
Enter the Sum d : 9
{ 1 2 6 }
{ 1 8 }
```

Lab Program 11a

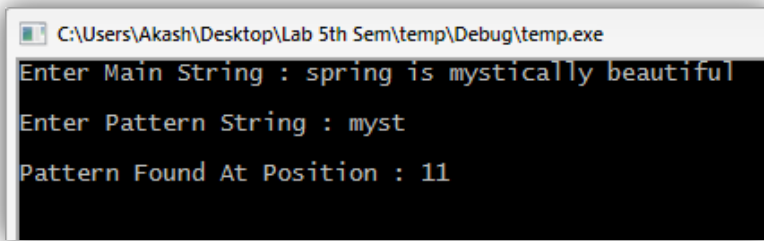
```
1  /**
2   Lab Program 11a
3   Horspool String Matching - Space n Time
4   Author SkyK0G
5   */
6
7   #include <iostream>
8   #include <conio.h>
9   #include <string.h>
10
11  using namespace std;
12
13  class HP {
14
15      char mainstr[50],pattern[50];
16      int mlen,plen,table[256];
17
18      public:
19          void shftable();
20          int matcher();
21          void input();
22  };
23
24  void main()
25  {
26      HP obj;
27
28      obj.input();
29      int pos=obj.matcher();
30
31      if(pos==-1)
32          cout<<"\nPattern Not Found";
33      else
34          cout<<"\nPattern Found At Position : "<<pos+1;
35
36      getch();
37  }
38
39  void HP::input()
40  {
41      cout<<"Enter Main String : ";
42      cin.getline(mainstr,50);
43
44      cout<<"\nEnter Pattern String : ";
45      cin.getline(pattern,50);
46  }
47
```

```

48 void HP::shftable()
49 {
50     plen=strlen(pattern);
51
52     for(int i=0;i<128;i++)
53         table[i]=plen;
54
55     for(int i=0;i<plen-1;i++)
56         table[pattern[i]]=plen-i-1;
57 }
58
59 int HP::matcher()
60 {
61     int i,k;
62
63     shftable();
64
65     mlen=strlen(mainstr);
66     plen=strlen(pattern);
67     i=plen-1;
68
69     while(i<mlen)
70     {
71         k=0;
72         while(k<plen && mainstr[i-k]==pattern[plen-1-k])
73             k++;
74         if(k==plen)
75             return i-plen+1;
76         i=i+table[mainstr[i]];
77     }
78     return -1;
79 }

```

Output

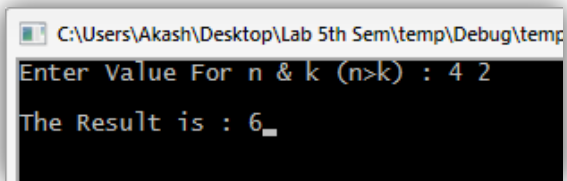


```
C:\Users\Akash\Desktop\Lab 5th Sem\temp\Debug\temp.exe
Enter Main String : spring is mystically beautiful
Enter Pattern String : myst
Pattern Found At Position : 11
```


Lab Program 11b

```
1  /**
2   Lab Program 11b
3   Binomial Coefficient - Dynamic Programming
4   Author SkyKOG
5   */
6
7   #include <iostream>
8   #include <conio.h>
9
10  #define MAX 10
11
12  using namespace std;
13
14  int bincoeff(int,int);
15
16  void main()
17  {
18      int n,k;
19
20      cout<<"Enter Value For n & k (n>k) : ";
21      cin>>n>>k;
22
23      int res=bincoeff(n,k);
24
25      cout<<"\nThe Result is : "<<res;
26
27      getch();
28  }
29
30  int bincoeff(int n,int k)
31  {
32      int c[MAX][MAX];
33
34      for(int i = 0 ; i <= n ; i++)
35          for(int j = 0 ; j <= k ; j++)
36              if(!j || i == j)
37                  c[i][j]=1;
38              else
39                  c[i][j]=c[i-1][j-1]+c[i-1][j];
40
41      return c[n][k];
42  }
```

Output



```
C:\Users\Akash\Desktop\Lab 5th Sem\temp\Debug\temp
Enter Value For n & k (n>k) : 4 2
The Result is : 6_
```

Lab Program 12

```
1  /**
2  Lab Program 12
3  Prim's Algorithm - Greedy
4  Author SkyKOG
5  */
6
7  #include <iostream>
8  #include <conio.h>
9
10 using namespace std;
11
12 #define MAX 10
13
14 int prim(int [MAX][MAX],int,int);
15
16 void main()
17 {
18     int a[MAX][MAX],source,n,i,j,m;
19
20     cout<<"Enter Number of Vertex : ";
21     cin>>n;
22
23     cout<<"\nEnter Cost Matrix : \n\n";
24     for(i=1;i<=n;i++)
25     {
26         for(j=1;j<=n;j++)
27             cin>>a[i][j];
28     }
29
30     cout<<"\nEnter Source : ";
31     cin>>source;
32     cout<<"\n";
33
34     m=prim(a,source,n);
35
36     cout<<"\nCost : "<<m;
37
38     getch();
39 }
```

```

41 int prim(int cost[MAX][MAX],int source,int n)
42 {
43     int i,j,sum=0,visited[MAX],cmp[MAX],vertex[MAX],min,u,v;
44
45     for(i=1;i<=n;i++)
46     {
47         vertex[i]=source;
48         visited[i]=0;
49         cmp[i]=cost[source][i];
50     }
51     visited[source]=1;
52     for(i=1;i<=n-1;i++)
53     {
54         min=999;
55         for(j=1;j<=n;j++)
56             if(!visited[j] && cmp[j]<min)
57             {
58                 min=cmp[j];
59                 u=j;
60             }
61         visited[u]=1;
62         sum=sum+cmp[u];
63         cout<<vertex[u]<<" -> "<<u<<" Sum = "<<sum<<"\n";
64         for(v=1;v<=n;v++)
65             if(!visited[v] && cost[u][v]<cmp[v])
66             {
67                 cmp[v]=cost[u][v];
68                 vertex[v]=u;
69             }
70     }
71     return sum;
72 }

```

Output

```
C:\Users\Akash\Desktop\Lab 5th Sem\temp\Debug\temp.exe
Enter Number of Vertex : 4
Enter Cost Matrix 0-No Edge 999-No Reach :
0 20 10 50
20 0 60 999
10 60 0 40
50 999 40 0

Enter Source : 1

1 -> 3 Sum = 10
1 -> 2 Sum = 30
3 -> 4 Sum = 70

Cost : 70_
```

Lab Program 13a

```
1  /**
2  Lab Program 13a
3  Floyd's All Pairs - Dynamic Programming
4  Author SkyKOG
5  */
6
7  #include<iostream>
8  #include<conio.h>
9
10 #define MAX 50
11
12 using namespace std;
13
14 class Floyds
15 {
16     int a[MAX][MAX],n;
17
18     public:
19         void costmat();
20         void floyd();
21         int min(int,int);
22         void display();
23 };
24
25 void Floyds::costmat()
26 {
27     cout<<"Enter Number of Nodes : ";
28     cin>>n;
29
30     cout<<"\nEnter The Cost Adjacency Matrix 0-Self 999-No Reach : \n\n";
31     for (int i=1;i<=n;i++)
32         for (int j=1;j<=n;j++)
33             cin>>a[i][j];
34 }
35
36 void main()
37 {
38     Floyds obj;
39
40     obj.costmat();
41     obj.floyd();
42     obj.display();
43
44     getch();
45 }
```

```

47 void Floyd::floyd()
48 {
49     for(int k = 1 ; k <= n ; k++)
50         for (int i = 1 ; i <= n ; i++)
51             for (int j = 1 ; j <= n ; j++)
52                 a[i][j] = min(a[i][j], (a[i][k] + a[k][j]));
53 }
54
55 void Floyd::display()
56 {
57     int i;
58
59     cout<<"\nSolution for Floyd's All Pairs Shortest Path :\n\n";
60     for (i=1;i<=n;i++)
61     {
62         for (int j=1;j<=n;j++)
63             cout<<a[i][j]<<" ";
64         cout<<"\n";
65     }
66 }
67
68 int Floyd::min(int a,int b)
69 {
70     return (a<b)?a:b;
71 }

```

Output

```
C:\Users\Akash\Desktop\Lab 5th Sem\temp\Debug\temp.exe
Enter Number of Nodes : 4
Enter The Cost Adjacency Matrix 0-Self 999-No Reach :
0 999 3 999
2 0 999 999
999 7 0 1
6 999 999 0

Solution for Floyd's All Pairs Shortest Path :
0 10 3 4
2 0 5 6
7 7 0 1
6 16 9 0
```


Lab Program 13b

```
1  /**
2   Lab Program 13b
3   Marshall's Transitive Closure - Dynamic Programming
4   Author SkyKOG
5   */
6
7   #include<iostream>
8   #include<conio.h>
9
10  #define MAX 50
11
12  using namespace std;
13
14  class Warshalls
15  {
16      int a[MAX][MAX],n;
17
18      public:
19          void costmat();
20          void wtc();
21          void display();
22  };
23
24  void Warshalls::costmat()
25  {
26      cout<<"Enter Number of Nodes : ";
27      cin>>n;
28
29      cout<<"\nEnter The Cost Adjacency Matrix (0-No Edge 1-Edge)\n\n";
30      for (int i=1;i<=n;i++)
31          for (int j=1;j<=n;j++)
32              cin>>a[i][j];
33  }
34
35  void main()
36  {
37      Warshalls obj;
38
39      obj.costmat();
40      obj.wtc();
41      obj.display();
42
43      getch();
44  }
```

```

46 void Warshalls::wtc()
47 {
48     for(int k = 1 ;k <= n ; k++)
49         for (int i = 1 ; i <= n ; i++)
50             for (int j = 1 ; j <= n ; j++)
51                 if(a[i][k] && a[k][j])
52                     a[i][j]=1;
53 }
54
55 void Warshalls::display()
56 {
57     int i;
58
59     cout<<"\nSolution for Warshall's Transitive Closure :\n\n";
60     for (i=1;i<=n;i++)
61     {
62         for (int j=1;j<=n;j++)
63             cout<<a[i][j]<<" ";
64         cout<<"\n";
65     }
66 }

```

Output

```
C:\Users\Akash\Desktop\Lab 5th Sem\temp\Debug\temp.exe
Enter Number of Nodes : 4
Enter The Cost Adjacency Matrix (0-No Edge 1-Edge)
0 1 0 0
0 0 1 0
0 0 0 1
0 0 0 0

Solution for Warshall's Transitive Closure :
0 1 1 1
0 0 1 1
0 0 0 1
0 0 0 0
```

Lab Program 14

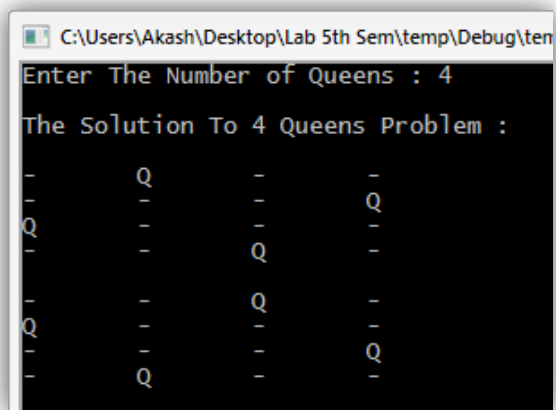
```
1  /**
2   Lab Program 14
3   N-Queens - Backtracking
4   Author SkyKOG
5   */
6
7   #include<iostream>
8   #include<math.h>
9   #include<stdlib.h>
10  #include <conio.h>
11
12  #define MAX 20
13
14  using namespace std;
15
16  void display(int [],int);
17  bool place(int [],int);
18  void nqueens(int );
19
20  void main()
21  {
22      int n;
23      cout<<"Enter The Number of Queens : ";
24      cin>>n;
25      cout<<"\nThe Solution To "<<n<<" Queens Problem :\n\n";
26      nqueens(n);
27      getch();
28  }
29
30  void nqueens(int n)
31  {
32      int x[MAX];
33      int k;
34
35      x[0]=-1;
36      k=0;
37
38      while(k>=0)
39      {
40          x[k]=x[k]+1;
41          while(x[k]<n&&!place(x,k))
42              x[k]=x[k]+1;
43          if(x[k]<n)
44              if(k==n-1)
45                  display(x,n);
46              else
47                  x[++k]=-1;
48          else
49              k--;
50      }
51  }
```

```

53 bool place(int x[],int k)
54 {
55     int i;
56
57     for(i=0;i<k;i++)
58         if(x[i]==x[k] || abs(x[i]-x[k])==abs(i-k))
59             return false;
60     return true;
61 }
62
63 void display(int x[],int n)
64 {
65     char chessb[MAX][MAX];
66     int i,j;
67
68     for(i=0;i<n;i++)
69         for(j=0;j<n;j++)
70             chessb[i][j]='-';
71
72     for(i=0;i<n;i++)
73         chessb[i][x[i]]='Q';
74
75     for(i=0;i<n;i++)
76     {
77         for(j=0;j<n;j++)
78             cout<<chessb[i][j]<<"\t";
79         cout<<"\n";
80     }
81     cout<<"\n";
82 }

```

Output



```
C:\Users\Akash\Desktop\Lab 5th Sem\temp\Debug\ten
Enter The Number of Queens : 4
The Solution To 4 Queens Problem :
-      Q      -      -
-      -      -      Q
Q      -      -      -
-      -      Q      -
-      -      Q      -
Q      -      -      Q
-      Q      -      -
```

Ya this is a bonus topic ...here we'll see how to calculate time taken for the sorting algorithms to sort huge numbersit's needed since we need to plot graphs for rate of growth ...

Here we will take the example of the simple selection sort ...the same can be applied to all other algorithms.

First increase the MAX size... take it whatever u want ... 1000, 50000, 99999 ...

```
11 | #define MAX 50000
```

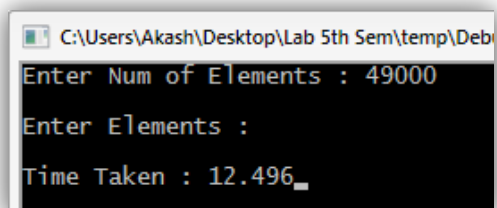
Next Disable taking input of numbers ... u don't wanna input 50k numbers do u ;)let the computer auto gen for u ☺.....u prolly need to include <stdlib.h>

```
25 | cout<<"\nEnter Elements : ";
26 | for(i = 0 ; i < n ; i++)
27 |     //cin>>a[i];
28 |     a[i]=rand();
```

Finally disable output of sorted numbers ...u don't wanna wait while the comp displays 50000 nums :P

```
34 | /*cout<<"\nThe Sorted Elements Are : ";
35 | for(i = 0 ; i < n ; i++)
36 |     cout<<a[i]<<" ";*/
```

Done !!! When u run the prog now u should get the output as follows:



```
C:\Users\Akash\Desktop\Lab 5th Sem\temp\Debu
Enter Num of Elements : 49000
Enter Elements :
Time Taken : 12.496_
```

Now for other sizes just change the MAX limit and enter whatever num as input size u want :D ...

Basically this works as follows:

First u create 2 clock variables ... one for noting the start time & the other stop time Just like a stopwatch 😊....

```
20 | clock_t start,end;
```

So now just like how one uses a stopwatch before and after the race ... store values in the clock variables “immediately” BEFORE & AFTER the block of code u want to measure time for ...

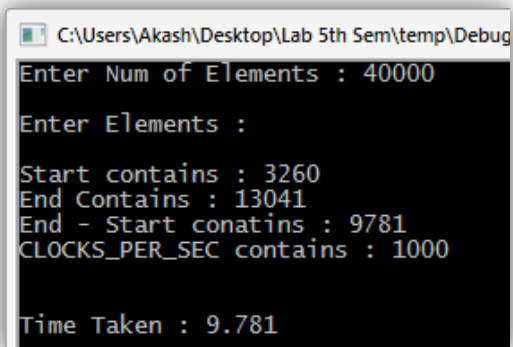
So in this example we use the variables before and after the selection sort starts and ends :

```
30 | start=clock();  
31 | selsort(a, n);  
32 | end=clock();
```

so start contains the snapshot of the current system clock pulse and the end contains the number of pulses elapsed with reference to start ... when u finally divide the subtracted value by the `CLOCKS_PER_SEC` u get time elapsed in seconds ...

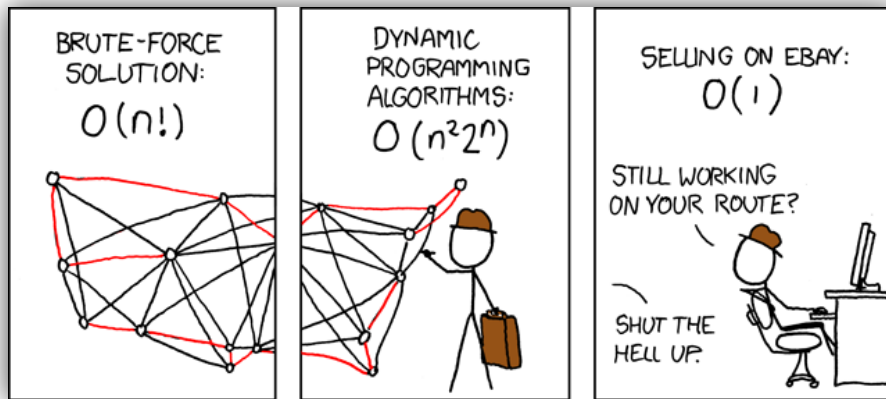
```
42 | cout<<"\n\nTime Taken : "<<(end-start)/(double)CLOCKS_PER_SEC;
```

Here is a sample run for one test :



```
C:\Users\Akash\Desktop\Lab 5th Sem\temp\Debug  
Enter Num of Elements : 40000  
Enter Elements :  
Start contains : 3260  
End Contains : 13041  
End - Start conatins : 9781  
CLOCKS_PER_SEC contains : 1000  
Time Taken : 9.781
```

Please note time calculation is system and current system load dependent so obviously will vary on different systems by a small margin ;)



(PS: Donald E Knuth is the world's Greatest Algorithms scientistbetter known as "Father of Algorithms")