

# ADA LAB RECORD-1

DIVYASHREE K

1BM19CS054

CSE-4A

Question 1: Write a recursive program to a.  
Solve Towers-of-Hanoi problem b. To find GCD

CODE: #include<stdio.h>

```
void merge(int b[],int c[],int a[],int  
nOfB,int nOfC){
```

```
    int i=0,j=0,k=0;
```

```
    // printf("Printing the Divided array  
B\n");
```

```
    // for(i=0;i<nOfB;i++){
```

```
        // printf("%d\t",b[i]);
```

```
// }  
  
// printf("\n");  
  
// printf("Printing the Divided array  
C\n");  
  
// for(i=0;i<nOfC;i++){  
//     printf("%d\t",c[i]);  
// }  
  
// printf("\n");  
  
// i=0;j=0;k=0;  
while(i<nOfB&& j<nOfC){  
    if(b[i]<=c[j]){  
        a[k]=b[i];  
        i++;  
    }else{  
        a[k]=c[j];
```

```
        j++;
    }

    k++;
}

//printf("Values of i=%d and
j=%d\n",i,j);

if(i==nOfB){
    for(j;j<nOfC;j++){
        a[k]=c[j];

        k++;
    }
}

else{
    for(i;i<nOfC;i++){
        a[k]=b[i];
```

```
        k++;
    }
}

// printf("Printing the Divided array
A\n");

// for(i=0;i<k;i++){
//     printf("%d\t",a[i]);
// }

// printf("\n");
}

void inputArray(int a[],int n){
    int i;
    for(i=0;i<n;i++){
        scanf("%d",&a[i]);
    }
}
```

```
}
```

```
int main(){
```

```
    int n1,n2,i,n,median;
```

```
    printf("Enter the number of  
elements in array 1\n");
```

```
    scanf("%d",&n1);
```

```
    printf("Enter the sort array 1\n");
```

```
    int b[n1];
```

```
    inputArray(b,n1);
```

```
    printf("Enter the number of  
elements in array 2\n");
```

```
    scanf("%d",&n2);
```

```
    printf("Enter the sort array 2\n");
```

```
    int c[n2];
```

```
    inputArray(c,n2);
```

```
n=n1+n2;

int a[n];

merge(b,c,a,n1,n2);

for(i=0;i<n1+n2;i++){
    printf("%d\t",a[i]);
}

printf("\n");

if(n%2==1){
    median=(n+1)/2;
    printf("Median is %d\n",a[median-
1]);
}

else {
    median=(n)/2;
```

```
        printf("Median is
%f\n", (float)(a[median-
1]+a[median])/2);

    }

    return 0;

}
```

OUTPUT:

```
Enter the number of elements in the array:
5
Enter 1 for linear search and 2 for binary search:
1
Enter the elements of the array :
2 3 6 8 9
Enter the element to be searched :
8
Element is found at the 3 index in the array.
```

```
Enter the number of elements in the array:
5
Enter 1 for linear search and 2 for binary search:
2
Enter the elements of the array in asscending order :
3
5
9
11
13
Enter the element to be seached :
5
Element is found at the 1 index in the array.
```

**Question 2:** Implement Recursive Binary search and Linear search and determine the time required to search an element. Repeat the experiment for different values of N and plot a graph of the time taken versus N

**CODE: a) BINARY SEARCH:**

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include <time.h>
```



```
int binarySearch(int arr[], int l, int r, int
x)
{
    if (r >= l) {
        int mid = l + (r - l) / 2;
        if (arr[mid] == x)
            return mid;
        if (arr[mid] > x)
            return binarySearch(arr, l,
mid - 1, x);
        return binarySearch(arr, mid + 1,
r, x);
    }
    return -1;
}
```

```
int main(void)
{
    int n,i,key;
    float start,end;

    printf("Enter the number of
elements\n");

    scanf("%d",&n);

    int arr[n];

    for(i=0;i<n;i++)
        arr[i]=rand();

    printf("Enter the element to
search\n");

    scanf("%d",&key);

    start=clock();
```

```
    int result = binarySearch(arr, 0, n -  
1, key);  
  
    end=clock();  
  
    if(result== -1)  
  
        printf("Element is not present in  
array");  
  
    else  
  
        printf("Element is present at index  
%d",result);  
  
        printf("\n Time taken to sort %d  
numbers is %f Secs",n, (((double)(end-  
start))/CLOCKS_PER_SEC));  
  
    return 0;  
  
}
```

OUTPUT:

```

Enter the number of elements
50
Enter 1 for linear search and 2 for binary search
1
1804289383 846930886 1681692777 1714636915 1957747793 424238335 719885386 1649760492 596516649 1189641421 1025202362 1350490027 7
83368690 1102520059 2044897763 1967513926 1365180540 1540383426 304089172 1303455736 35005211 521595368 294702567 1726956429 3364
65782 861021530 278722862 233665123 2145174067 468703135 1101513929 1801979802 1315634022 635723050 1369133069 1125898167 1059961
393 2089018456 628175011 1656478042 1131176229 1653377373 859484421 1914544919 608413784 756898537 1734575198 1973594324 14979831
5 2038664370 Enter the element to search
2038664370
Element is present at index 49
Time taken to sort 50 numbers is 0.000004 Secs

...Program finished with exit code 0
Press ENTER to exit console.

```

```

Enter the number of elements
50
Enter 1 for linear search and 2 for binary search
1
1804289383 846930886 1681692777 1714636915 1957747793 424238335 719885386 1649760492 596516649 1189641421 1025202362 1350490027 7
83368690 1102520059 2044897763 1967513926 1365180540 1540383426 304089172 1303455736 35005211 521595368 294702567 1726956429 3364
65782 861021530 278722862 233665123 2145174067 468703135 1101513929 1801979802 1315634022 635723050 1369133069 1125898167 1059961
393 2089018456 628175011 1656478042 1131176229 1653377373 859484421 1914544919 608413784 756898537 1734575198 1973594324 14979831
5 2038664370 Enter the element to search
2336
Element is not present in array
Time taken to sort 50 numbers is 0.000003 Secs

...Program finished with exit code 0
Press ENTER to exit console.

```

CODE: b)LINEAR SEARCH

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include <time.h>
```

```
int linearSearch(int arr[], int l, int n, int
x)
{
    if (l >= n) {
        if (arr[l] == x)
            return l;
        return linearSearch(arr, l + 1, n,
x);
    }
    return -1;
}
```

```
int main(void)
{
    int n,i,key;
```

```
float start,end;

    printf("Enter the number of
elements\n");

    scanf("%d",&n);

    int arr[n];

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

        arr[i]=rand();

    printf("Enter the element to
search\n");

    scanf("%d",&key);

    start=clock();

    int result = linearSearch(arr, 0, n - 1,
key);

    end=clock();

    if(result==-1)
```

```
        printf("Element is not present in  
array");  
  
    else  
  
        printf("Element is present at index  
%d",result);  
  
        printf("\n Time taken to sort %d  
numbers is %f Secs",n, (((double)(end-  
start))/CLOCKS_PER_SEC));  
  
    return 0;  
  
}
```

OUTPUT:

```

Enter the number of elements
50
Enter 1 for linear search and 2 for binary search
2
35005211 149798315 233665123 278722862 294702567 304089172 336465782 424238335 468703135 521595368 596516649 608413784 628175011
635723058 719885386 756898537 783368690 846930886 859484421 861021530 1025202362 1059961393 1101513929 1102520059 1125898167 1131
176229 1189641421 1303455736 1315634022 1350490027 1365180540 1369133069 1540383426 1649760492 1653377373 1656478042 1681692777 1
714636915 1726956429 1734575198 1801979802 1804289383 1914544919 1957747793 1967513926 1973594324 2038664370 2044897763 208901845
6 2145174067 Enter the element to search
1303455736
Element is present at index 27
Time taken to sort 50 numbers is 0.000002 Secs

...Program finished with exit code 0
Press ENTER to exit console.

```

```

Enter the number of elements
50
Enter 1 for linear search and 2 for binary search
2
35005211 149798315 233665123 278722862 294702567 304089172 336465782 424238335 468703135 521595368 596516649 608413784 628175011
635723058 719885386 756898537 783368690 846930886 859484421 861021530 1025202362 1059961393 1101513929 1102520059 1125898167 1131
176229 1189641421 1303455736 1315634022 1350490027 1365180540 1369133069 1540383426 1649760492 1653377373 1656478042 1681692777 1
714636915 1726956429 1734575198 1801979802 1804289383 1914544919 1957747793 1967513926 1973594324 2038664370 2044897763 208901845
6 2145174067 Enter the element to search
23569
Element is not present in array
Time taken to sort 50 numbers is 0.000002 Secs

...Program finished with exit code 0
Press ENTER to exit console.

```

Question 3: Sort a given set of N integer elements using Selection Sort technique and compute its time taken. Run the program for different values of N and record the time taken to sort.

CODE: #include <stdio.h>

#include <time.h>

#include <stdlib.h>



```
void delay(int milli_seconds)
{
    clock_t start_time = clock();
    while (clock() < start_time
+milli_seconds)
        ;
}
```

```
void selectionSort(int arr[], int n)
{
    int i, j, min,t;
    for (i = 0; i < n-1; i++)
    {
        min = i;
        for (j = i+1; j < n; j++)
```

```
{  
    delay(10);  
    if (arr[j] < arr[min])  
        min = j;  
}  
t=arr[i];  
arr[i]=arr[min];  
arr[min]=t;  
}  
}
```

```
void printSelection(int arr[], int size)  
{
```

```
int i;  
for (i=0; i < size; i++)  
    printf("%d ", arr[i]);  
printf("\n");  
}
```

```
int main()  
{  
    int n;  
    clock_t start, end;  
    double cpu_time_used;  
    printf("Enter the size of the  
array\n");  
    scanf("%d",&n);
```

```
int arr[n];

printf("The elements of the
array:\n");

for(int i=0;i<n;i++)
arr[i]=rand();

printSelection(arr, n);

printf("\n");

start = clock();

selectionSort(arr, n);

end = clock();

cpu_time_used = ((double) (end -
start)) / CLOCKS_PER_SEC;

printf("Sorted array: \n");

printSelection(arr, n);

printf("\n");
```

```

    printf("TIME FOR FUNCTION
EXECUTION is %f", cpu_time_used);

    return 0;
}

```

OUTPUT:

```

How many numbers u are going to enter?: 10

Time taken to sort 10 numbers is 0.000004 Secs
424238335,596516649,719885386,846930886,1189641421,1649760492,1681692777,1714636915,1804289383,1957747793,

...Program finished with exit code 0
Press ENTER to exit console.

```

```

How many numbers u are going to enter?: 50

Time taken to sort 50 numbers is 0.000014 Secs
35005211,149798315,233665123,278722862,294702567,304089172,336465782,424238335,468703135,521595368,596516649,608413784,628175011,
635723058,719885386,756898537,783368690,846930886,859484421,861021530,1025202362,1059961393,1101513929,1102520059,1125898167,1131
176229,1189641421,1303455736,1315634022,1350490027,1365180540,1369133069,1540383426,1649760492,1653377373,1656478042,1681692777,1
714636915,1726956429,1734575198,1801979802,1804289383,1914544919,1957747793,1967513926,1973594324,2038664370,2044097763,208901845
6,2145174067,

...Program finished with exit code 0
Press ENTER to exit console.

```

Question 4: Write program to do the following:

- Print all the nodes reachable from a given starting node in a digraph using BFS method.
- Check whether a given graph is connected or not using DFS method.

CODE:

a)BFS:

```
#include<stdio.h>
```

```
int a[10][10],vis[10],n;
```

```
void bfs(int v)
```

```
{
```

```
    int q[10],f=1,r=1,u,i;
```

```
    q[r]=v;
```

```
    vis[v]=1;
```

```
    while(f<=r)
```

```
    {
```

```
        u=q[f];
```

```
        printf("%d ",u);
```

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

```
{  
    if(a[u][i]==1&&vis[i]==0)  
    {  
        vis[i]=1;  
        r=r+1;  
        q[r]=i;  
  
    }  
}  
f=f+1;  
}  
}
```

```
void main()
{
    int i,j,src;

    printf("enter the no of vertices\n");
    scanf("%d",&n);
    printf("enter the adjacency\n");
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++)
        {
            scanf("%d",&a[i][j]);
        }
        vis[i]=0;
    }
}
```



```
}
```

```
printf("enter the source vertex\n");
```

```
scanf("%d",&src);
```

```
printf("nodes reachable from source  
vertex are\n");
```

```
bfs(src);
```

```
}
```

OUTPUT:a)

```
Enter the number of vertices:4

Enter graph data in matrix form:
0 1 0 0
0 0 0 1
0 1 0 0
0 0 0 0

Enter the starting vertex:1

Time taken for bfs is 0.000002
The node which are reachable are:
2      4

...Program finished with exit code 255
Press ENTER to exit console.█
```

b)DFS:

```
#include<stdio.h>

#include<conio.h>

int a[20][20],reach[20],n;

void dfs(int v) {
    int i;
    reach[v]=1;
    for (i=1;i<=n;i++)
        if(a[v][i] && !reach[i]) {
            printf("\n %d->%d",v,i);
            dfs(i);
        }
}

void main() {
    int i,j,count=0;
```

```
printf("\n Enter the total number of  
vertices:");
```

```
scanf("%d",&n);
```

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

```
    reach[i]=0;
```

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

```
        a[i][j]=0;
```

```
}
```

```
printf("\n Enter the adjacency  
matrix:\n");
```

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

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

```
        scanf("%d",&a[i][j]);
```

```
dfs(1);
```

```
printf("\n");  
for (i=1;i<=n;i++) {  
    if(reach[i])  
        count++;  
}  
if(count==n)  
    printf("\n Graph is connected");  
else  
    printf("\n Graph is not  
connected");  
}  
OUTPUT:b)
```

```
Enter number of vertices:4

Enter the adjacency matrix:
0 1 0 0
0 0 0 1
0 1 0 0
0 0 0 0

1->2
2->4
Time taken for dfs is 0.000018

Graph is not connected

...Program finished with exit code 255
Press ENTER to exit console.█
```

Question 5: Sort a given set of N integer elements using Insertion Sort technique and compute its time taken.

CODE: #include <stdio.h>

#include <time.h>

#include <stdlib.h>

```
void delay(int milli_seconds)
{
    clock_t start_time = clock();
    while (clock() < start_time
+milli_seconds)
        ;
}
```

```
void insertionSort(int arr[], int n)
{
    int i, j, min,t;
    for (i = 0; i < n-1; i++)
    {
        min = i;
        for (j = i+1; j < n; j++)
```

```
{  
    delay(10);  
  
    if (arr[j] < arr[min])  
        min = j;  
}  
t=arr[i];  
arr[i]=arr[min];  
arr[min]=t;  
}  
}
```

```
void printArray(int arr[], int size)
```

```
{  
    int i;  
    for (i=0; i < size; i++)  
        printf("%d ", arr[i]);  
    printf("\n");  
}
```

```
int main()  
{  
    int n;  
    clock_t start, end;  
    double cpu_time_used;  
    printf("Enter the size of the  
array\n");
```



```
scanf("%d",&n);

int arr[n];

printf("The elements of the
array:\n");

for(int i=0;i<n;i++)
arr[i]=rand();

printArray(arr, n);

printf("\n");

start = clock();

insertionSort(arr, n);

end = clock();

cpu_time_used = ((double) (end -
start)) / CLOCKS_PER_SEC;

printf("Sorted array: \n");

printArray(arr, n);
```

```
printf("\n");

printf("TIME FOR FUNCTION
EXECUTION is %f", cpu_time_used);

return 0;

}
```

OUTPUT:

```
Enter the size of the list: 20
Unsorted Array: 4 7 18 16 14 16 7 13 10 2 3 8 11 20 4 7 1 7 13 17
Sorted Array:   1 2 3 4 4 7 7 7 7 8 10 11 13 13 14 16 16 17 18 20
Time taken: 0.000003
```

```
Enter the size of the list: 50
Unsorted Array: 34 37 28 16 44 36 37 43 50 22 13 28 41 10 14 27 41 27 23 37 12 19 18 30 33 31 13 24 18 36 30 3 23 9 20 18 44
7 12 43 30 24 22 20 35 38 49 25 16 21
Sorted Array:   3 7 9 10 12 12 13 13 14 16 16 18 18 18 19 20 20 21 22 22 23 23 24 24 25 27 27 28 28 30 30 30 31 33 34 35 36
36 37 37 37 38 41 41 43 43 44 44 49 50
Time taken: 0.000006
```

```

Enter the size of the list: 500
Unsorted Array: 384 387 278 416 294 336 387 493 150 422 363 28 191 60 264 427 41 427 173 237 212 369 68 430 283 31 363 124 68
136 430 303 23 59 70 168 394 457 12 43 230 374 422 420 285 38 199 325 316 371 414 27 92 481 457 374 363 171 497 282 306 426 85
328 337 6 347 230 314 358 125 396 83 46 315 368 435 365 44 251 88 309 277 179 289 85 404 152 255 400 433 61 177 369 240 13 22
7 87 95 40 296 71 435 379 468 102 98 403 318 493 153 257 302 281 287 442 366 190 445 120 441 230 32 118 98 272 482 176 210 428
68 357 498 354 87 466 307 184 220 125 29 372 233 330 4 20 271 369 209 216 341 150 297 224 119 246 347 452 422 56 380 489 265
229 342 351 194 1 35 265 125 415 488 357 244 492 228 366 360 437 433 52 438 229 276 408 475 122 359 396 30 238 236 294 319 429
144 12 429 30 277 405 444 264 114 39 107 341 405 319 129 189 370 418 418 497 325 244 471 184 491 500 273 226 145 91 6 140 455
287 170 83 43 465 198 8 356 305 349 112 123 329 300 344 247 69 341 423 312 311 106 302 162 231 379 306 321 237 445 127 23 466
209 417 283 259 425 138 63 125 101 37 453 400 380 51 469 72 474 132 382 431 434 395 161 164 200 482 400 497 460 274 314 169 1
91 96 427 467 85 341 91 185 377 43 437 108 446 257 180 419 388 413 349 173 160 10 337 211 343 88 207 302 214 373 322 256 320 1
00 222 405 440 312 441 168 206 229 128 151 485 159 421 225 423 270 397 82 131 85 293 473 173 351 126 386 223 300 141 43 399 21
4 299 191 25 91 210 82 320 337 233 156 495 5 380 270 274 277 351 256 361 143 80 385 494 206 122 68 5 114 462 255 327 260 445 2
03 203 7 285 22 343 369 29 190 373 409 459 499 37 309 254 249 304 334 134 149 391 255 68 247 369 30 1 47 289 298 250 491 304 3
4 364 498 254 393 187 126 153 497 476 189 158 230 437 461 415 422 461 305 29 28 51 249 57 403 295 198 200 44 40 3 429 404 1 18
2 148 39 160 152 36 135 340 193 216 128 5 130 50 465 286 430 344 336 178 401 239 472 450 290 368 489 293 296 244 145 330 391 1
83 341 42 70 327 233
Sorted Array: 1 1 1 3 4 5 5 5 6 6 7 8 10 12 12 13 20 22 23 23 25 27 28 28 29 29 29 30 30 30 31 32 34 35 36 37 37 38 39 39 40
40 41 42 43 43 43 44 44 46 47 50 51 51 52 56 57 59 60 61 63 68 68 68 68 68 69 70 70 71 72 80 82 82 83 83 85 85 85 85 87 87
88 88 91 91 91 92 95 96 98 98 100 101 102 106 107 108 112 114 114 118 119 120 122 122 123 124 125 125 125 126 126 127 128
128 129 130 131 132 134 135 136 138 140 141 143 144 145 145 148 149 150 150 151 152 152 153 153 156 158 159 160 160 161 162 1
64 168 168 169 170 171 173 173 173 176 177 178 179 180 182 183 184 184 185 187 189 189 190 190 191 191 191 193 194 198 198 199
200 200 203 203 206 206 207 209 209 210 210 211 212 214 214 216 216 220 222 223 224 225 226 227 228 229 229 229 230 230 230 2
30 231 233 233 233 236 237 237 238 239 240 244 244 244 246 247 247 249 249 250 251 254 254 255 255 255 256 256 257 257 259 260
264 264 265 265 270 270 271 272 273 274 274 276 277 277 277 278 281 282 283 283 285 285 286 287 287 289 289 290 293 293 294 2
94 295 296 296 297 298 299 300 300 302 302 303 304 304 305 305 306 306 307 309 309 311 312 312 314 314 315 316 318 319 319
320 320 321 322 325 325 327 327 328 329 330 330 334 336 336 337 337 337 340 341 341 341 341 341 342 343 343 344 344 347 347 3
49 349 351 351 351 354 356 357 357 358 359 360 361 363 363 363 364 365 366 366 368 368 369 369 369 369 369 370 371 372 373 373
374 374 377 379 379 380 380 380 382 384 385 386 387 387 388 391 391 393 394 395 396 396 397 399 400 400 400 401 403 403 404 4
04 405 405 405 408 409 413 414 415 415 416 417 418 418 419 420 421 422 422 422 422 423 423 425 426 427 427 427 428 429 429 429
430 430 430 431 433 433 434 435 435 437 437 437 438 440 441 441 442 444 445 445 445 446 450 452 453 455 457 457 459 460 461 4
61 462 465 465 466 466 467 468 469 471 472 473 474 475 476 481 482 482 485 488 489 489 491 491 492 493 493 494 495 497 497 497
497 498 498 499 500
Time taken: 0.000284

```

Question 6: Write program to obtain the Topological ordering of vertices in a given digraph

CODE: #include <stdio.h>

```

int main(){

    int i,j,k,n,a[10][10],indeg[10],flag[10],count=0;

    printf("Enter the no of vertices:\n");

```

```
scanf("%d",&n);
```

```
printf("Enter the adjacency matrix:\n");
```

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

```
    printf("Enter row %d\n",i+1);
```

```
    for(j=0;j<n;j++)
```

```
        scanf("%d",&a[i][j]);
```

```
}
```

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

```
    indeg[i]=0;
```

```
    flag[i]=0;
```

```
}
```

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

```
    for(j=0;j<n;j++)
```

```
        indeg[i]=indeg[i]+a[j][i];
```

```
printf("\nThe topological order is:");
```

```
while(count<n){
```

```
    for(k=0;k<n;k++){
```

```
        if((indeg[k]==0) && (flag[k]==0)){
```

```
            printf("%d ",(k+1));
```

```
            flag [k]=1;
```

```
        }
```

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

```
            if(a[i][k]==1)
```

```
                indeg[k]--;
```

```
        }
```

```
    }
```

```
    count++;
```

```
}
```

```
    return 0;  
}
```

**OUTPUT:**

**Question 7:** Implement Johnson Trotter algorithm to generate permutations

CODE: #include<stdio.h>

```
int LEFT_TO_RIGHT = 1;  
int RIGHT_TO_LEFT = 0;
```

```
int searchArr(int a[], int n, int mobile)  
{  
    for (int i = 0; i < n; i++)  
        if (a[i] == mobile)  
            return i + 1;  
}
```

```
}
```

```
int getMobile(int a[], int dir[], int n)
```

```
{
```

```
    int mobile_prev = 0, mobile = 0;
```

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

```
    {
```

```
        if (dir[a[i]-1] == 0 && i!=0)
```

```
        {
```

```
            if (a[i] > a[i-1] && a[i] > mobile_prev)
```

```
            {
```

```
                mobile = a[i];
```

```
                mobile_prev = mobile;
```

```
            }
```

```
        }
```

```
if (dir[a[i]-1] == 1 && i!=n-1)
{
    if (a[i] > a[i+1] && a[i] > mobile_prev)
    {
        mobile = a[i];
        mobile_prev = mobile;
    }
}
}
```

```
if (mobile == 0 && mobile_prev == 0)
    return 0;
else
    return mobile;
}
```

```
void swap(int a[], int pos_1, int pos_2)
```



```
{  
    int temp = a[pos_1];  
    a[pos_1] = a[pos_2];  
    a[pos_2] = temp;  
}
```

```
int printOnePerm(int a[], int dir[], int n)
```

```
{  
    int mobile = getMobile(a, dir, n);  
    int pos = searchArr(a, n, mobile);
```

```
    if (dir[a[pos - 1] - 1] == 0)  
        swap(a, pos-1, pos-2);
```

```
    else if (dir[a[pos - 1] - 1] == 1)  
        swap(a, pos, pos-1);
```

```
for (int i = 0; i < n; i++)  
{  
    if (a[i] > mobile)  
    {  
        if (dir[a[i] - 1] == 1)  
            dir[a[i] - 1] = 0;  
        else if (dir[a[i] - 1] == 0)  
            dir[a[i] - 1] = 1;  
    }  
}
```

```
for (int i = 0; i < n; i++)  
    printf("%d", a[i]);  
printf(" ");  
}
```

```
int fact(int n)
{
    int res = 1;
    for (int i = 1; i <= n; i++)
        res = res * i;
    return res;
}
```

```
void printPermutation(int n)
{
```

```
    int a[n];
```

```
    int dir[n];
```

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

```
{
```

```
    a[i] = i + 1;
```

```
    printf("%d", a[i]);
```

```
}
```

```
printf("\n");
```

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

```
    dir[i] = 0;
```

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

```
    printOnePerm(a, dir, n);
```

```
}
```

```
int main()
```

```
{  
    int n;  
    printf("Enter the value\n");  
    scanf("%d",&n);  
    printPermutation(n);  
    return 0;  
}
```

**OUTPUT:**

**Question 8:** Sort a given set of N integer elements using Merge Sort technique and compute its time taken. Run the program for different values of N and record the time taken to sort.

**CODE:** #include <stdio.h>

#include <stdlib.h>

#include<time.h>

void merge(int arr[], int l, int m, int r)

{

```
int i, j, k;
```

```
int n1 = m - l + 1;
```

```
int n2 = r - m;
```

```
int L[n1], R[n2];
```

```
for (i = 0; i < n1; i++)
```

```
    L[i] = arr[l + i];
```

```
for (j = 0; j < n2; j++)
```

```
    R[j] = arr[m + 1 + j];
```

```
i = 0;
```

```
j = 0;
```

```
k = l;
```

```
while (i < n1 && j < n2) {
```

```
    if (L[i] <= R[j]) {
```

```
    arr[k] = L[i];  
    i++;  
}  
else {  
    arr[k] = R[j];  
    j++;  
}  
k++;  
}
```

```
while (i < n1) {  
    arr[k] = L[i];  
    i++;  
    k++;  
}
```

```
while (j < n2) {  
    arr[k] = R[j];
```

```
        j++;  
        k++;  
    }  
}
```

```
void mergeSort(int arr[], int l, int r)  
{  
  
    if (l < r) {  
        int m = l + (r - l) / 2;  
        mergeSort(arr, l, m);  
        mergeSort(arr, m + 1, r);  
        merge(arr, l, m, r);  
    }  
}
```

```
void printArray(int A[], int size)  
{
```



```
int i;  
for (i = 0; i < size; i++)  
    printf("%d ", A[i]);  
printf("\n");  
}
```

```
int main()  
{  
    int arr_size,i;  
    clock_t start, end;  
    printf("enter the array size\n");  
    scanf("%d",&arr_size);  
    int arr[arr_size];  
    for(i=0;i<arr_size;i++)  
    {  
        arr[i]=rand()%1000;  
    }  
    printf("Given array is \n");
```

```
    printArray(arr, arr_size);  
    start=clock();  
    mergeSort(arr, 0, arr_size - 1);  
    for(i=0;i<8000000;i++);  
    end=clock();  
    printf("\nSorted array is \n");  
    printArray(arr, arr_size);  
    printf("\n Time taken to sort %d numbers is %f  
Secs",arr_size, (((double)(end-  
start))/CLOCKS_PER_SEC));  
    return 0;  
}
```

**OUTPUT:**

```
Enter the size of the list: 10
Unsorted Array: 4 7 8 6 4 6 7 3 10 2
Sorted Array:   2 3 4 4 6 6 7 7 8 10
Time taken: 0.000004
```

```
Enter the size of the list: 30
Unsorted Array: 14 17 28 26 24 26 17 13 10 2 3 8 21 20 24 17 1 7 23 17 12 9 28 10 3 21 3 14 8 26
Sorted Array:   1 2 3 3 3 7 8 8 9 10 10 12 13 14 14 17 17 17 17 20 21 21 23 24 24 26 26 26 28 28
Time taken: 0.000007
```

```
Enter the size of the list: 50
Unsorted Array: 34 37 28 16 44 36 37 43 50 22 13 28 41 10 14 27 41 27 23 37 12 19 18 30 33 31 13 24 18 36 30 3 23 9 20 18 44
7 12 43 30 24 22 20 35 38 49 25 16 21
Sorted Array:   3 7 9 10 12 12 13 13 14 16 16 18 18 18 19 20 20 21 22 22 23 23 24 24 25 27 27 28 28 30 30 30 31 33 34 35 36
36 37 37 37 38 41 41 43 43 44 44 49 50
Time taken: 0.000011
```

```
Enter the size of the list: 100
Unsorted Array: 84 87 78 16 94 36 87 93 50 22 63 28 91 60 64 27 41 27 73 37 12 69 68 30 83 31 63 24 68 36 30 3 23 59 70 68 9
4 57 12 43 30 74 22 20 85 38 99 25 16 71 14 27 92 81 57 74 63 71 97 82 6 26 85 28 37 6 47 30 14 58 25 96 83 46 15 68 35 65 4
4 51 88 9 77 79 89 85 4 52 55 100 33 61 77 69 40 13 27 87 95 40
Sorted Array:   3 4 6 6 9 12 12 13 14 14 15 16 16 20 22 22 23 24 25 25 26 27 27 27 27 28 28 30 30 30 30 31 33 35 36 36 37 37
38 40 40 41 43 44 46 47 50 51 52 55 57 57 58 59 60 61 63 63 63 64 65 68 68 68 68 69 69 70 71 71 73 74 74 77 77 78 79 81 82
83 83 84 85 85 85 87 87 87 88 89 91 92 93 94 94 95 96 97 99 100
Time taken: 0.000017
```

Question 9: Sort a given set of N integer elements using Quick Sort technique and compute its time taken

CODE: #include<stdio.h>

#include<time.h>

```
void quicksort(int number[25],int
first,int last){

    int i, j, pivot, temp;

    if(first<last){

        pivot=first;

        i=first;

        j=last;


        while(i<j){

            while(number[i]<=number[pivot]&& i<l
ast)

                i++;

            while(number[j]>number[pivot])

                j--;
```

```
    if(i<j){  
        temp=number[i];  
        number[i]=number[j];  
        number[j]=temp;  
    }  
}
```

```
temp=number[pivot];  
number[pivot]=number[j];  
number[j]=temp;  
quicksort(number,first,j-1);  
quicksort(number,j+1,last);  
  
}
```

```
}
```

```
int main(){  
    int i, count;  
    clock_t start, end;  
    printf("How many elements are u  
going to enter?: ");  
    scanf("%d",&count);  
    int number[count];  
    printf("Given array is \n");  
    for(i=0;i<count;i++){  
        number[i]=rand()%1000;  
        printf("%d ", number[i]);  
    }  
    start=clock();
```

```
quicksort(number,0,count-1);  
for(i=0;i<8000000;i++);  
end=clock();  
printf("\nOrder of Sorted elements:  
\n");  
for(i=0;i<count;i++)  
    printf("%d ",number[i]);  
printf("\n Time taken to sort %d  
numbers is %f Secs",count,  
(((double)(end-  
start))/CLOCKS_PER_SEC));  
return 0;  
}
```

OUTPUT:

```
Enter the size of the list: 10
Unsorted Array: 4 7 8 6 4 6 7 3 10 2
Sorted Array:   2 3 4 4 6 6 7 7 8 10
Time taken: 0.000003
```

```
Enter the size of the list: 30
Unsorted Array: 14 17 28 26 24 26 17 13 10 2 3 8 21 20 24 17 1 7 23 17 12 9 28 10 3 21 3 14 8 26
Sorted Array:   1 2 3 3 3 7 8 8 9 10 10 12 13 14 14 17 17 17 17 20 21 21 23 24 24 26 26 26 28 28
Time taken: 0.000005
```

```
Enter the size of the list: 50
Unsorted Array: 34 37 28 16 44 36 37 43 50 22 13 28 41 10 14 27 41 27 23 37 12 19 18 30 33 31 13 24 18 36 30 3 23 9 20 18 44
7 12 43 30 24 22 20 35 38 49 25 16 21
Sorted Array:   3 7 9 10 12 12 13 13 14 16 16 18 18 18 19 20 20 21 22 22 23 23 24 24 25 27 27 28 28 30 30 30 31 33 34 35 36
36 37 37 37 38 41 41 43 43 44 44 49 50
Time taken: 0.000007
```

```
Enter the size of the list: 100
Unsorted Array: 84 87 78 16 94 36 87 93 50 22 63 28 91 60 64 27 41 27 73 37 12 69 68 30 83 31 63 24 68 36 30 3 23 59 70 68 9
4 57 12 43 30 74 22 20 85 38 99 25 16 71 14 27 92 81 57 74 63 71 97 82 6 26 85 28 37 6 47 30 14 58 25 96 83 46 15 68 35 65 4
4 51 88 9 77 79 89 85 4 52 55 100 33 61 77 69 40 13 27 87 95 40
Sorted Array:   3 4 6 6 9 12 12 13 14 14 15 16 16 20 22 22 23 24 25 25 26 27 27 27 27 27 28 28 30 30 30 30 31 33 35 36 36 37 37
38 40 40 41 43 44 46 47 50 51 52 55 57 57 58 59 60 61 63 63 63 64 65 68 68 68 68 69 69 70 71 71 73 74 74 77 77 78 79 81 82
83 83 84 85 85 85 87 87 88 89 91 92 93 94 94 95 96 97 99 100
Time taken: 0.000013
```

Question 10: Sort a given set of N integer elements using Heap Sort technique and compute its time taken.

CODE: #include<stdio.h>

void heapSort(int h[],int n){



```
int i,k,v,heap,j;
for(i=n/2;i>=1;i--){
    k=i;
    v=h[k];
    heap=0;
    while(heap==0 && 2*k<=n){
        j=2*k;
        if(j<n){
            if(h[j]<h[j+1])
                j=j+1;
        }
        if(v>=h[j]){
            heap=1;
        }else{
```

```
        h[k]=h[j];
        k=j;
    }
}
h[k]=v;
}
}

int main(){
    int n,i;
    printf("Enter no of elements\n");
    scanf("%d",&n);
    int h[n];
    printf("Enter the heap in array
representation\n");
    for(i=1;i<=n;i++)
```

```
        scanf("%d",&h[i]);  
heapSort(h,n);  
printf("Printing the sorted heap\n");  
for(i=1;i<=n;i++){  
    printf("%d\t",h[i]);  
    if(i%5==0)  
        printf("\n");  
}  
return 0;  
}
```

OUTPUT:

```
How many numbers u are going to enter?: 5
3 1 2 0 3

Time taken to sort 5 numbers is 0.000004 Secs
Sorted array is
0 1 2 3 3
```

```
How many numbers u are going to enter?: 10
3 6 7 5 3 5 6 2 9 1

Time taken to sort 10 numbers is 0.000003 Secs
Sorted array is
1 2 3 3 5 5 6 6 7 9
```

```
How many numbers u are going to enter?: 50
83 36 27 15 43 35 36 42 49 21 12 27 40 9 13 26 40 26 22 36 11 18 17 29 32 30 12 23 17 35 29 2
22 8 19 17 43 6 11 42 29 23 21 19 34 37 48 24 15 20

Time taken to sort 50 numbers is 0.000021 Secs
Sorted array is
2 6 8 9 11 11 12 12 13 15 15 17 17 17 18 19 19 20 21 21 22 22 23 23 24 26 26 27 27 29 29 29
30 32 33 34 35 35 36 36 36 37 40 40 42 42 43 43 48 49
```

Question 11: Implement Warshall's algorithm using dynamic programming.

CODE: #include<stdio.h>

#include<stdlib.h>

```
void transitiveClosure(int A[10][10],int  
n)
```

```
{
```

```
    int path[n][n];
```

```
    int i,j,k;
```

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

```
    {
```

```
        for(j=0;j<n;j++)
```

```
        {
```

```
            path[i][j] = A[i][j];
```

```
        }
```

```
    }
```

```
    for(k=0;k<n;k++)
```

```
{  
    for(i=0;i<n;i++)  
    {  
        for(j=0;j<n;j++)  
        {  
            if(path[i][j] != 1 && (path[i][k]  
&& path[k][j]))  
                path[i][j] = 1;  
        }  
    }  
}
```

```
for(i=0;i<n;i++)  
{  
    for(j=0;j<n;j++)
```

```
    {  
        printf("\t%d",path[i][j]);  
    }  
    printf("\n");  
}  
}
```

```
int main()  
{  
    int A[10][10];  
    int i,j;  
    int n;  
    printf("\nEnter The Number Of  
Vertices : ");  
    scanf("%d",&n);
```

```
    printf("\nEnter The Graph Data in  
the form of Adacency Matrix : \n");  
    for(i=0;i<n;i++)  
    {  
        for(j=0;j<n;j++)  
        {  
            scanf("%d",&A[i][j]);  
        }  
    }  
    transitiveClosure(A,n);  
    return 0;  
}
```

OUTPUT:



```
Enter no of vertices: 4
Enter adjacency matrix:
0 1 0 0
0 0 0 1
0 0 0 0
1 0 1 0

Time taken is 0.000004
Transitive Closure:
1 1 1 1
1 1 1 1
0 0 0 0
1 1 1 1
```

Question 12: Implement 0/1 Knapsack problem using dynamic programming.

CODE: #include<stdio.h>

```
int max(int a, int b) { return (a > b)? a :
b; }
```

```
int knapSack(int W, int wt[], int val[],
int n)
{
    int i, w;
    int K[n+1][W+1];

    for (i = 0; i <= n; i++)
    {
        for (w = 0; w <= W; w++)
        {
            if (i==0 || w==0)
                K[i][w] = 0;
            else if (wt[i-1] <= w)
                K[i][w] = max(val[i-1] + K[i-1][w-wt[i-1]], K[i-1][w]);
            else
                K[i][w] = K[i-1][w];
        }
    }
    return K[n][W];
}
```

```
    else
```

```
        K[i][w] = K[i-1][w];
```

```
    }
```

```
}
```

```
return K[n][W];
```

```
}
```

```
int main()
```

```
{
```

```
    int i, n, val[20], wt[20], W;
```

```
    printf("Enter number of items:");
```

```
    scanf("%d", &n);
```

```
    printf("Enter value and weight of  
items:\n");  
  
    for(i = 0; i < n; ++i){  
        scanf("%d%d", &val[i], &wt[i]);  
    }  
  
    printf("Enter size of knapsack:");  
    scanf("%d", &W);  
  
    printf("%d", knapSack(W, wt, val, n));  
    return 0;  
}
```

OUTPUT:

```
Enter the number of items : 3
Enter the values :
25
15
40
Enter the weights :
2
1
3
Enter the weight of the bag :
3

The value is : 40
Time taken is 0.000025
```

Question 13: Implement All Pair Shortest paths problem using Floyd's algorithm.

CODE: #include<stdio.h>

```
int a[10][10],d[10][10],n;
```

```
void floyds();  
int min(int ,int);  
int main()  
{  
    printf("Enter no of vertices\n");  
    scanf("%d",&n);  
    printf("Enter cost adjacency matrix  
:\n");  
    for(int i=1;i<=n;i++)  
    {  
        for(int j=1;j<=n;j++)  
            scanf("%d",&a[i][j]);  
    }  
    floyds();  
    printf("Distance matrix :\n");
```

```
for(int i=1;i<=n;i++)
{
    for(int j=1;j<=n;j++)
        printf("%d ",d[i][j]);
    printf("\n");
}
}
```

```
void floyds()
{
    for(int i=1;i<=n;i++)
    {
        for(int j=1;j<=n;j++)
            d[i][j]=a[i][j];
    }
}
```

```
}  
for(int k=1;k<=n;k++)  
{  
    for(int i=1;i<=n;i++)  
    {  
        for(int j=1;j<=n;j++)  
        {  
  
            d[i][j]=min(d[i][j],d[i][k]+d[k][j]);  
  
        }  
    }  
}  
}  
  
int min(int a,int b)  
{
```



```
if(a<b)
    return a;
else
    return b;
}
```

OUTPUT:

```
No of vertices: 4
Enter Weight matrix(-1 if there is no edge):
0 2 3 7
-1 0 1 -1
-1 -1 0 5
-1 -1 -1 0

Time taken is 0.000004Distance matrix:
0    2    3    7
INF 0    1    6
INF INF 0    5
INF INF INF 0
```

**Question 14:** Find Minimum Cost Spanning Tree of a given undirected graph using Prim's algorithm

**CODE:** #include<stdio.h>

int minKey(int key[], int mstSet[],int n)

{

int min = 100, min\_index;

int v;

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

if (mstSet[v] == 0 && key[v] < min)

min = key[v], min\_index = v;

return min\_index;

}

```
int printMST(int parent[10], int
graph[10][10],int n)
{
    int i;
    printf("Edge \tWeight\n");
    for ( i = 1; i < n; i++)
        printf("%d - %d \t%d \n", parent[i],
i, graph[i][parent[i]]);
}
```

```
void primMST(int graph[10][10],int n)
{

    int parent[n];
    int key[n];
```

```
int mstSet[n];
```

```
int i,count,v,u;
```

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

```
    key[i] = 100, mstSet[i] = 0;
```

```
key[0] = 0;
```

```
parent[0] = -1;
```

```
for ( count = 0; count < n - 1;  
count++) {
```

```
u = minKey(key, mstSet,n);
```

```
mstSet[u] = 1;
```

```
for (v = 0; v < n; v++)
```

```
    if (graph[u][v] && mstSet[v] == 0  
    && graph[u][v] < key[v])
```

```
        parent[v] = u, key[v] =  
graph[u][v];
```

```
}
```

```
printMST(parent, graph,n);
```

```
}
```

```
int main()
{
    int graph[10][10];
    int i,j,n;
    printf("Enter number of nodes\n");
    scanf("%d",&n);
    printf("Enter adjacency matrix\n");
    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
            scanf("%d",&graph[i][j]);
    }
    primMST(graph,n);
}
```

```
    return 0;
}
```

OUTPUT:

```
Enter number of nodes
5
Enter adjacency matrix
0 2 0 6 0
2 0 3 8 5
0 3 0 0 7
6 8 0 0 9
0 5 7 9 0
Edge      Weight
0 - 1      2
1 - 2      3
0 - 3      6
1 - 4      5
```

**Question 15:** Find Minimum Cost Spanning Tree of a given undirected graph using Kruskals algorithm.

CODE: #include <stdio.h>

#include <stdlib.h>

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

int

min,mincost=0,cost[9][9],parent[9];

int find(int i)

{

while(parent[i])

i=parent[i];

return i;

}

int uni(int i,int j)

{

if(i!=j)

{



```
    parent[j]=i;
    return 1;
}
return 0;
}

void main()
{
    printf("\nEnter the no. of vertices:");
    scanf("%d",&n);

    printf("\nEnter the cost adjacency
matrix:\n");

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

```
scanf("%d",&cost[i][j]);  
if(cost[i][j]==0)  
cost[i][j]=999;  
}  
}  
  
printf("The edges of Minimum Cost  
Spanning Tree are\n");  
while(ne < n)  
{  
for(i=1,min=999;i<=n;i++)  
{  
for(j=1;j <= n;j++)  
{  
if(cost[i][j] < min)  
{
```

```
min=cost[i][j];  
a=u=i;  
b=v=j;  
}  
}  
}  
u=find(u);  
v=find(v);  
if(uni(u,v))  
{  
    printf("%d edge (%d,%d)  
=%d\n",ne++,a,b,min);  
    mincost +=min;  
}  
cost[a][b]=cost[b][a]=999;
```

```
    }  
  
    printf("\n\tMinimum cost =  
%d\n",mincost);  
  
}
```

OUTPUT:

```
Enter the no. of vertices:3  
  
Enter the cost adjacency matrix:  
0 10 10  
10 0 0  
0 20 0  
The edges of Minimum Cost Spanning Tree are  
1 edge (1,2) =10  
2 edge (1,3) =10  
  
Minimum cost = 20
```

**Question 16:** From a given vertex in a weighted connected graph, find shortest paths to other vertices using Dijkstra's algorithm

**CODE:** #include <stdio.h>

```
int minDistance(int dist[], int
sptSet[],int V)
{
    int min = 999, min_index;
    int v;
    for ( v = 0; v < V; v++)
        if (sptSet[v] == 0 && dist[v] <= min)
            min = dist[v], min_index = v;

    return min_index;
}
```

```

int printSolution(int dist[],int V )
{
    int i;

    printf("Vertex \t\t Distance from\nSource\n");

    for (i = 0; i < V; i++)
        printf("%d \t\t %d\n", i, dist[i]);
}

void dijkstra(int graph[10][10], int
src,int V)
{
    int dist[V];

    int i,count,u,v;

    int sptSet[V];

    for ( i = 0; i < V; i++)
        dist[i] = 999, sptSet[i] = 0;

```

```
dist[src] = 0;
```

```
for ( count = 0; count < V - 1;  
count++) {
```

```
    u = minDistance(dist, sptSet,V);
```

```
    sptSet[u] = 1;
```

```
    for (v = 0; v < V; v++)
```

```
        if (!sptSet[v] && graph[u][v] &&  
dist[u] != 999
```

```
        && dist[u] + graph[u][v] <  
dist[v]))
```

```
        dist[v] = dist[u] + graph[u][v];
    }

    printSolution(dist,V);
}

int main()
{
    int i,j,V;

    int graph[10][10];
    printf("Enter number of vertices\n");
    scanf("%d",&V);
    printf("Enter adjacency matrix\n");
```



```
for(i=0;i<V;i++)  
{  
    for(j=0;j<V;j++)  
        scanf("%d",&graph[i][j]);  
}  
  
dijkstra(graph, 0,V);  
  
return 0;  
}
```

OUTPUT:

```

Enter the number of vertices : 5
Enter the matrix :
0 0 0 7 0
3 0 4 0 0
0 0 0 0 6
0 2 5 0 0
0 0 0 4 0

Vertex    Distance    Path
0 -> 1      9          0 3 1
0 -> 2     12          0 3 2
0 -> 3      7          0 3
0 -> 4     18          0 3 2 4

```

**Question 17:** Implement “Sum of Subsets” using Backtracking. “Sum of Subsets” problem: 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.

**CODE:** `#include<stdio.h>`

```
#include<stdlib.h>
```

```
int x[1000],w[1000],d,count=0;
```

```
void subset(int cs,int k,int r)
```

```
{
```

```
    int i;
```

```
    x[k]=1;
```

```
    if(cs+w[k]==d)
```

```
    {
```

```
        printf("\nSubset %d\n",++count);
```

```
        for(i=0;i<=k;i++)
```

```
            if(x[i]==1)
```

```
                printf("%d\t",w[i]);
```

```
}  
else if(cs+w[k]+w[k+1]<=d)  
    subset(cs+w[k],k+1,r-w[k]);  
if(cs+r-w[k]>=d && cs+w[k]<=d)  
{  
    x[k]=0;  
    subset(cs,k+1,r-w[k]);  
}  
}
```

```
int main()  
{  
    int i,n,sum=0;
```

```
printf("Enter the no. of elements: ");  
scanf("%d",&n);  
  
printf("Enter the elements in  
ascending order:\n");  
  
for(i=0;i<n;i++)  
    scanf("%d",&w[i]);  
  
printf("Enter the sum : ");  
scanf("%d",&d);  
  
for(i=0;i<n;i++)  
    sum=sum+w[i];  
  
if(sum<d)  
{  
    printf("No solution\n");  
    exit(0);  
}
```

```
subset(0,0,sum);  
if(count==0)  
{  
    printf("No solution\n");  
    exit(0);  
}  
return 0;  
}
```

OUTPUT:

```
Enter the no. of elements: 6
Enter the elements in ascending order:
1 2 3 4 5 6
Enter the sum : 9
```

```
Subset 1
1      2      6
Subset 2
1      3      5
Subset 3
2      3      4
Subset 4
3      6
Subset 5
4      5
```

Question 18: Implement “N-Queens Problem” using Backtracking.

CODE: #include<stdio.h>

#include<stdlib.h>

#include<math.h>

void nqueens(int);

```
int place(int[],int);
```

```
void printsolution(int,int[]);
```

```
void main()
```

```
{
```

```
    int n;
```

```
    printf("Enter the no.of queens: ");
```

```
    scanf("%d",&n);
```

```
    nqueens(n);
```

```
}
```

```
void nqueens(int n)
```

```
{
```

```
    int x[10],count=0,k=1;
```



```

x[k]=0;
while(k!=0)
{
    x[k]=x[k]+1;
    while(x[k]<=n&&(!place(x,k)))
        x[k]=x[k]+1;
    if(x[k]<=n)
    {
        if(k==n)
        {
            count++;
            printf("\nSolution
%d\n",count);
            printsolution(n,x);
        }
    }
}

```

```
        else
        {
            k++;
            x[k]=0;
        }
    }
    else
    {
        k--; //backtracking
    }
}
return;
}
```

```
int place(int x[],int k)
{
    int i;
    for(i=1;i<k;i++)
        if(x[i]==x[k] || (abs(x[i]-
x[k]))==abs(i-k))
            return 0;
    return 1;
}
```

```
void printsolution(int n,int x[])
{
    int i,j;
    char c[10][10];
    for(i=1;i<=n;i++)
```

```
{  
    for(j=1;j<=n;j++)  
        c[i][j]='X';  
}  
for(i=1;i<=n;i++)  
    c[i][x[i]]='Q';  
for(i=1;i<=n;i++)  
{  
    for(j=1;j<=n;j++)  
    {  
        printf("%c\t",c[i][j]);  
    }  
    printf("\n");  
}
```

}

OUTPUT:

```
Enter the size of the board : 4
```

```
0  0  1  0
```

```
1  0  0  0
```

```
0  0  0  1
```

```
0  1  0  0
```

