

1. Write a program to return all the possible subsets for a given integer array. Return the solution in any order.

Input nums= [1,2,3]

Output : [[], [1], [2], [3], [1,2], [1,3], [2,3], [1,2,3]]

Program :

```
#include <stdio.h>

char string[50], n;

void subset(int, int, int);

int main()
{
    int i, len;

    printf("Enter the len of main set : ");
    scanf("%d", &len);

    printf("Enter the elements of main set : ");
    scanf("%s", string);

    n = len;

    printf("The subsets are :\n");

    for (i = 1; i <= n; i++)
        subset(0, 0, i);
}

void subset(int start, int index, int num_sub)
{
    int i, j;

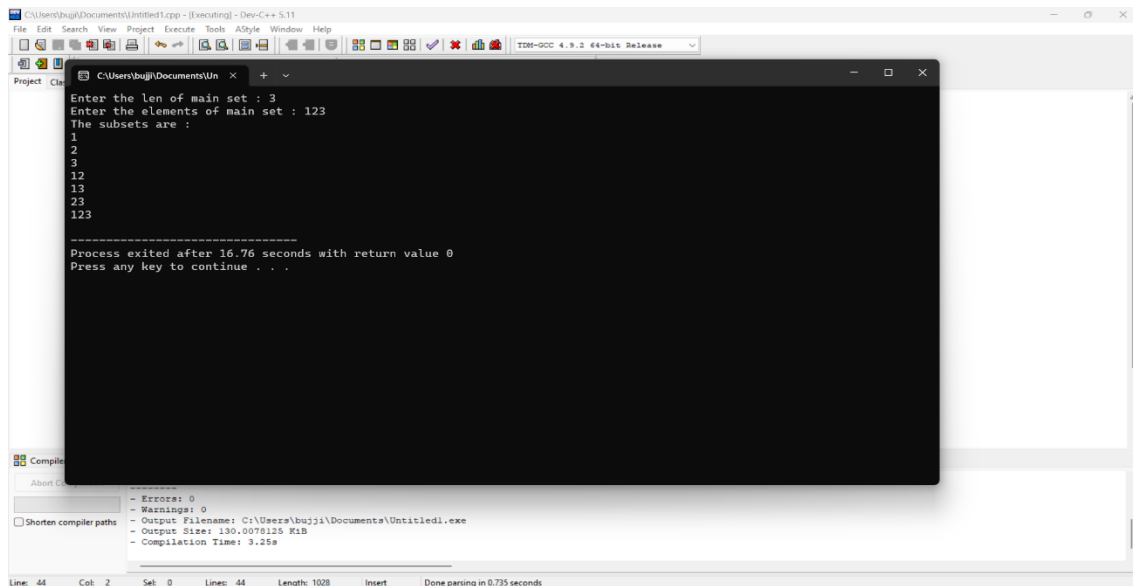
    if (index - start + 1 == num_sub)
    {
        if (num_sub == 1)
```

```

{
    for (i = 0; i < n; i++)
        printf("%c\n", string[i]);
}
else
{
    for (j = index; j < n; j++)
    {
        for (i = start; i < index; i++)
            printf("%c", string[i]);
        printf("%c\n", string[j]);
    }
    if (start != n - num_sub)
        subset(start + 1, start + 1, num_sub);
}
}
else
{
    subset(start, index + 1, num_sub);
}
}

```

Output :



2. Write a program to perform sum of subsets problem using backtracking and estimate time complexity. Identify the test cases.

A. Set (s) = (6, 2, 8, 1, 5) sum is 9 B. Set (s) = (6, -4, 7, -1, 5, 2, 8, 1,) sum is 10

Program :

```
#include <stdio.h>

#include <stdlib.h>

static int total_nodes;

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

void subset_sum(int s[], int t[], int s_size, int t_size, int sum, int ite, int const target_sum){
    total_nodes++;

    if (target_sum == sum) {
        printValues(t, t_size);
        subset_sum(s, t, s_size, t_size - 1, sum - s[ite], ite + 1, target_sum);
        return;
    }
}
```

```

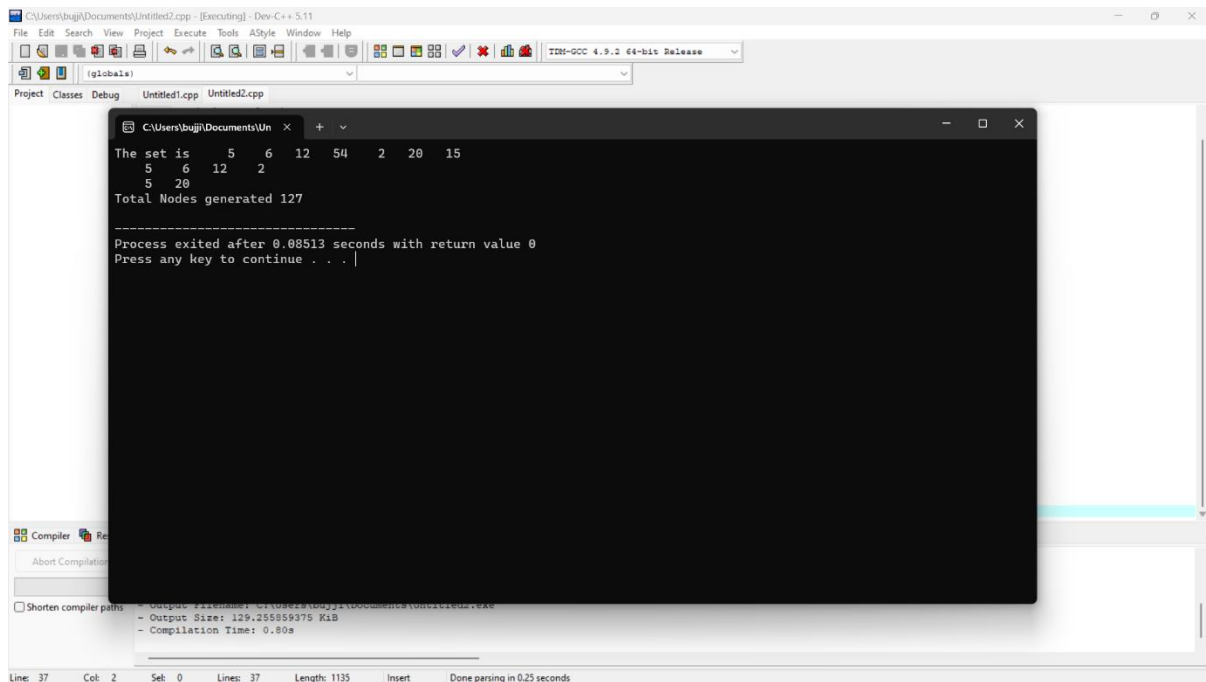
else {
    for (int i = ite; i < s_size; i++) {
        t[t_size] = s[i];
        subset_sum(s, t, s_size, t_size + 1, sum + s[i], i + 1, target_sum);
    }
}
}

void generateSubsets(int s[], int size, int target_sum){
    int* tuple_vector = (int*)malloc(size * sizeof(int));
    subset_sum(s, tuple_vector, size, 0, 0, 0, target_sum);
    free(tuple_vector);
}

int main(){
    int set[] = { 5, 6, 12 , 54, 2 , 20 , 15 };
    int size = sizeof(set) / sizeof(set[0]);
    printf("The set is ");
    printValues(set , size);
    generateSubsets(set, size, 25);
    printf("Total Nodes generated %d\n", total_nodes);
    return 0;
}

```

Output :

A screenshot of a C++ IDE window titled "C:\Users\buji\Documents\Untitled2.cpp - [Executing] - Dev-C++ 5.11". The IDE shows a project named "Untitled2.cpp" with a "Project" tab selected. A black console window is open, displaying the following output:

```
The set is 5 6 12 54 2 20 15
5 6 12 2
5 20
Total Nodes generated 127

-----
Process exited after 0.08513 seconds with return value 0
Press any key to continue . . .
```

The IDE's status bar at the bottom shows "Line: 37 Col: 2 Sel: 0 Lines: 37 Length: 1135 Insert Done parsing in 0.25 seconds".

3. Determine an optimal tour in a weighted, directed graph. The weights are nonnegative numbers. The inputs are weighted, directed graph, and n , the number of vertices in the graph. The graph is represented by a two-dimensional array W , which has both its rows and columns indexed from 1 to n , where $W[i][j]$ is the weight on the edge from the i th vertex to the j th vertex. Write a program for travelling salesman problem using dynamic programming for the below given graph.

Program :

```
#include <stdio.h>

#include <stdbool.h>

#define MAX 20

#define INF 99999

int n, d[MAX][MAX], x[MAX];

int best_tour_length = INF, tour_length[MAX];
```

```

void backtrack(int curr_pos) {
    int i;
    if (curr_pos == n) {
        tour_length[curr_pos] = d[x[n - 1]][x[0]];
        int tour = 0;
        for (i = 0; i < n; i++) tour += tour_length[i];
        if (tour < best_tour_length) best_tour_length = tour;
        return;
    }
    for (i = 0; i < n; i++) {
        if (x[i] == -1) {
            x[i] = curr_pos;
            tour_length[curr_pos] = d[x[curr_pos - 1]][i];
            backtrack(curr_pos + 1);
            x[i] = -1;
        }
    }
}

int main() {
    int i, j;
    printf("Enter the number of cities: ");
    scanf("%d", &n);
    printf("Enter the distance matrix:\n");
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++) {
            scanf("%d", &d[i][j]);
            x[i] = -1;
        }
    x[0] = 0;

```

```

backtrack(1);

printf("The minimum tour length is: %d\n", best_tour_length);

return 0;
}

```

Output :

```

C:\Users\bajji\Documents\Untitled3.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
[Icons] (globals)
Project Classes Debug
Untitled1.cpp Untitled2.cpp [*]Untitled3.cpp
10 void backtrack(int curr_pos) {
11     int i;
12     if (curr_pos == n) {
13         tour_length[curr_pos] = d[x[n-1]][x[0]];
14         int tour = 0;
15     }
16 }
17
18
19 Enter the number of cities: 3
20 Enter the distance matrix:
21 1 2 3
22 3 4 5
23 6 7 8
24
25 The minimum tour length is: 7
26
27 -----
28 Process exited after 24.93 seconds with return value 0
29 Press any key to continue . . .
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
Compiler Resources
Abort Compilation
Shorten compiler paths
Line: 44 Col: 1 Sel: 0 Lines: 44 Length: 978 Insert Done parsing in 0.094 seconds

```

4.The n-queens puzzle is the problem of placing n queens on an n x n chessboard such that no two queens attack each other. Given an integer n, return all distinct solutions to the n-queens puzzle. You may return the answer in any order. Write a program for the same.

Program :

```

#include <stdio.h>

#include <stdbool.h>

#define N 8

int col[N];

bool check(int row) {

    int i;

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

        if (col[i] == col[row] ||

```

```

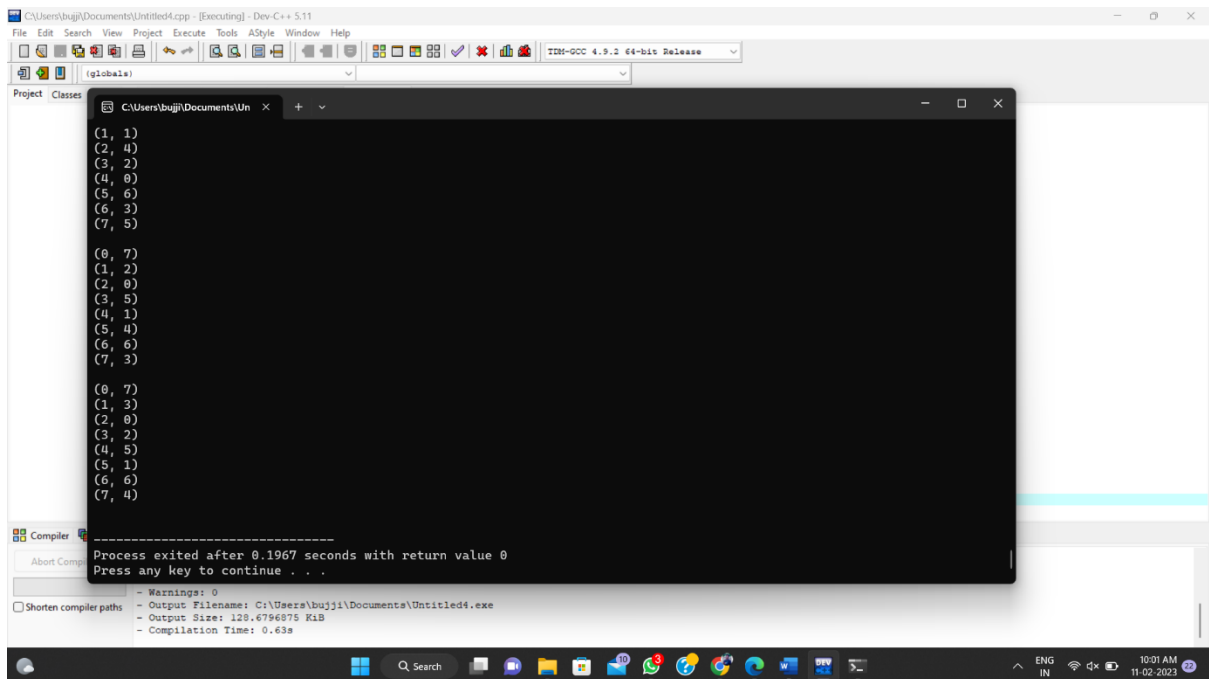
        row - i == col[row] - col[i] ||
        row - i == col[i] - col[row])
    return false;
return true;
}

void backtrack(int row) {
    int i;
    if (row == N) {
        for (i = 0; i < N; i++) printf("(%d, %d)\n", i, col[i]);
        printf("\n");
        return;
    }
    for (i = 0; i < N; i++) {
        col[row] = i;
        if (check(row)) backtrack(row + 1);
    }
}

int main() {
    backtrack(0);
    return 0;
}

```


Output :



```
C:\Users\bujji\Documents\Untitled4.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
(globals)
Project Classes
C:\Users\bujji\Documents\Un x + -
(1, 1)
(2, 4)
(3, 2)
(4, 0)
(5, 6)
(6, 3)
(7, 5)
(0, 7)
(1, 2)
(2, 0)
(3, 5)
(4, 1)
(5, 4)
(6, 6)
(7, 3)
(0, 7)
(1, 3)
(2, 0)
(3, 2)
(4, 5)
(5, 1)
(6, 6)
(7, 4)
-----
Process exited after 0.1967 seconds with return value 0
Press any key to continue . . .
- Warnings: 0
- Output Filename: C:\Users\bujji\Documents\Untitled4.exe
- Output Size: 120.6796875 KiB
- Compilation Time: 0.63s
Shorten compiler paths
10:01 AM 11-02-2023
```

5. Write a program to perform Minimum spanning tree using greedy techniques and estimate time complexity for the given set of values.

Program :

```
#include <stdio.h>

#include <limits.h>

#define V 5

int minKey(int key[], int mstSet[]) {
    int min = INT_MAX, min_index;

    int v;

    for (v = 0; v < V; v++)
        if (mstSet[v] == 0 && key[v] < min)
            min = key[v], min_index = v;

    return min_index;
}
```

```

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

void primMST(int graph[V][V]) {
    int parent[V];
    int key[V], i, v, count;
    int mstSet[V];
    for (v = 0; v < V; v++)

        if (graph[u][v] && mstSet[v] == 0 && graph[u][v] < key[v])
            parent[v] = u, key[v] = graph[u][v];
    }
    printMST(parent, V, graph);
}

int main() {
    2  3
    (0)--(1)--(2)
    |  /\  |
    6| 8/  \5 |7
    | /   \ |
    (3)----- (4)
    9      */

    int graph[V][V] = { { 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 }, };
    primMST(graph);
    return 0;
}

```

```
}
```

Output :

```
Edge  Weight
0 - 1   2
1 - 2   3
0 - 3   6
1 - 4   5

Process returned 0 (0x0)   execution time : 0.035 s
Press any key to continue.
```

6. Write a C program for binary search tree and find the time complexity

Program :

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

struct node
{
    int data;
    struct node*left;
    struct node*right;
}*root=NULL,*newnode;

struct node*create(struct node*root,int ele)
{
    if(root==NULL)
    {
        newnode=(struct node*)malloc(sizeof(struct node));
        newnode->data=ele;
        newnode->left=NULL;
        newnode->right=NULL;
        return(newnode);
    }
}
```

```

        else if(ele>root->data)
            root->right=create(root->right,ele);
        else if(ele<root->data)
            root->left=create(root->left,ele);
        return(root);
    }

void inorder(struct node *root)
{
    if(root!=NULL)
    {
        inorder(root->left);
        printf("%d\t",root->data);
        inorder(root->right);
    }

}

void preorder(struct node *root)
{
    if(root!=NULL)
    {
        printf("%d\t",root->data);
        preorder(root->left);
        preorder(root->right);
    }

}

void postorder(struct node *root)
{
    if(root!=NULL)
    {

```

```

        postorder(root->left);
        postorder(root->right);
        printf("%d\t",root->data);
    }
}

int main()
{
    int choice;
    while(1)
    {
        printf("\nMAIN MENU\n");
        printf("\n1.CREATE\n");
        printf("\n2.INORDER\n");
        printf("\n3.PREORDER\n");
        printf("\n4.POSTORDER\n");
        printf("\n5.EXIT\n");
        printf("\nENTER THE CHOICE:\t");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
                int ele;
                printf("ENTER THE ELEMENT:");
                scanf("%d",&ele);
                root=create(root,ele);
                break;
            case 2:
                inorder(root);
                break;

```

```

        case 3:
            preorder(root);
            break;

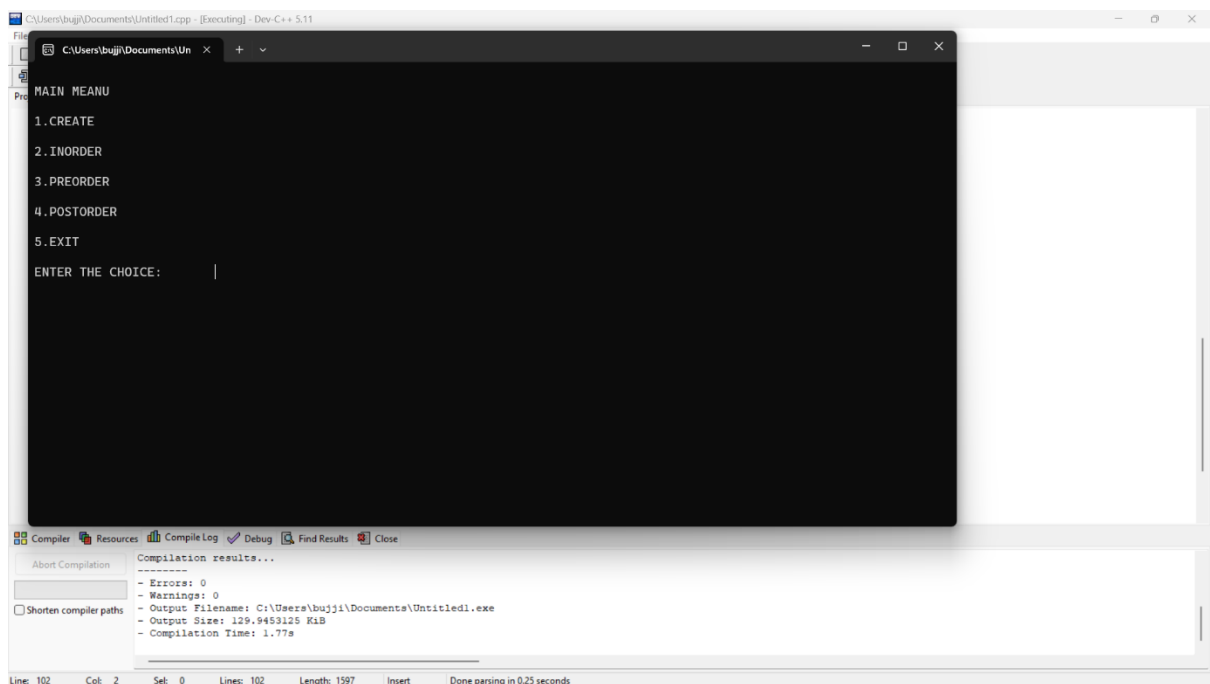
        case 4:
            postorder(root);
            break;

        case 5:
            exit(0);
            break;

        default:
            printf("\nWRONG CHOICE\n");
            break;
    }
}
}

```

Output :



The screenshot shows a C++ IDE window titled "C:\Users\bujj\Documents\Untitled1.cpp - [Executing] - Dev-C++ 5.11". The main window displays the output of a program, which is a menu for a binary tree traversal application. The menu options are: 1.CREATE, 2.INORDER, 3.PREORDER, 4.POSTORDER, and 5.EXIT. Below the menu, it says "ENTER THE CHOICE:" followed by a cursor. The bottom panel shows the "Compilation results..." window, which reports: Errors: 0, Warnings: 0, Output Filename: C:\Users\bujj\Documents\Untitled1.exe, Output Size: 129.9453125 KiB, and Compilation Time: 1.77s. The status bar at the bottom indicates "Line: 102, Col: 2, Set: 0, Lines: 102, Length: 1597, Insert, Done parsing in 0.25 seconds".

```

C:\Users\bujj\Documents\Untitled1.cpp - [Executing] - Dev-C++ 5.11
C:\Users\bujj\Documents\Un
MAIN MENU
1.CREATE
2.INORDER
3.PREORDER
4.POSTORDER
5.EXIT
ENTER THE CHOICE: |

Compiler
Resources
Compile Log
Debug
Find Results
Close
Compilation results...
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\bujj\Documents\Untitled1.exe
- Output Size: 129.9453125 KiB
- Compilation Time: 1.77s
Shorten compiler paths
Line: 102 Col: 2 Set: 0 Lines: 102 Length: 1597 Insert Done parsing in 0.25 seconds

```

7. Let there be N workers and N jobs. Any worker can be assigned to perform any job, incurring some cost that may vary depending on the work-job assignment. It is required to perform all jobs by assigning exactly one worker to each job and exactly one job to each agent in such a way that the total cost of the assignment is minimized. Write a program to solve an assignment problem for the given data sets using branch and bound.

	Job 1	Job 2	Job 3	Job 4
Person A	12	8	9	10
Person B	11	10	10	9
Person C	9	11	8	12
Person D	11	9	23	7

Program :

```
#include <stdbool.h>
```

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

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

```
typedef struct Job {
```

```
    char id;
```

```
    int dead;
```

```
    int profit;
```

```
} Job;
```

```
int compare(const void* a, const void* b)
```

```
{
```

```
    Job* temp1 = (Job*)a;
```

```
    Job* temp2 = (Job*)b;
```

```
        return (temp2->profit - temp1->profit);
    }
```

```
int min(int num1, int num2)
{
    return (num1 > num2) ? num2 : num1;
}
```

```
void printJobScheduling(Job arr[], int n)
{
    qsort(arr, n, sizeof(Job), compare);

    int result[n];
    bool slot[n];

    for (int i = 0; i < n; i++)
        slot[i] = false;

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

        for (int j = min(n, arr[i].dead) - 1; j >= 0; j--) {

            if (slot[j] == false) {
                result[j] = i;
            }
        }
    }
}
```



```

        slot[j] = true;
        break;
    }
}

for (int i = 0; i < n; i++)
    if (slot[i])
        printf("%c ", arr[result[i]].id);
}

int main()
{
    Job arr[] = { { 'a', 12, 8, 9, 10 },
                  { 'b', 11, 10, 10, 9 },
                  { 'c', 9, 11, 8, 12 },
                  { 'd', 11, 9, 23, 7 } };

    int n = sizeof(arr) / sizeof(arr[0]);

    printf(
        "Following is maximum profit sequence of jobs \n");

    printJobScheduling(arr, n);

    return 0;
}

```

C:\Users\Admin\Documents\daa28-job.exe

Following is maximum profit sequence of jobs

a d b c

Process returned 0 (0x0) execution time : 0.055 s

Press any key to continue.