

1)Write a C program to merge sort using divide and Conquer

```
#include <stdio.h>

#include <stdlib.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++;
    }
}
```

DAY-2 PROGRAMS

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

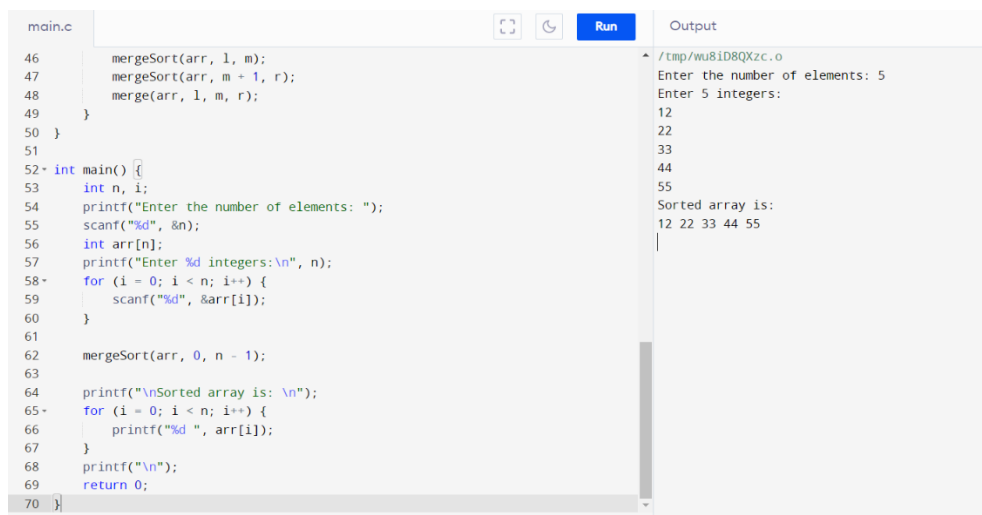
while (j < n2) {
    arr[k] = R[j];
    j++;
    k++;
}
}

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);
    }
}

int main() {
    int n, i;
    printf("Enter the number of elements: ");
    scanf("%d", &n);
    int arr[n];
    printf("Enter %d integers:\n", n);
    for (i = 0; i < n; i++)
```

DAY-2 PROGRAMS

```
{  
    scanf("%d", &arr[i]);  
}  
  
mergeSort(arr, 0, n - 1);  
  
printf("\nSorted array is: \n");  
for (i = 0; i < n; i++)  
{  
    printf("%d ", arr[i]);  
}  
printf("\n");  
return 0;  
}
```



```
main.c  
46     mergeSort(arr, l, m);  
47     mergeSort(arr, m + 1, r);  
48     merge(arr, l, m, r);  
49 }  
50 }  
51  
52 int main() {  
53     int n, i;  
54     printf("Enter the number of elements: ");  
55     scanf("%d", &n);  
56     int arr[n];  
57     printf("Enter %d integers:\n", n);  
58     for (i = 0; i < n; i++) {  
59         scanf("%d", &arr[i]);  
60     }  
61  
62     mergeSort(arr, 0, n - 1);  
63  
64     printf("\nSorted array is: \n");  
65     for (i = 0; i < n; i++) {  
66         printf("%d ", arr[i]);  
67     }  
68     printf("\n");  
69     return 0;  
70 }
```

Output

```
/tmp/wu81D8QXzc.o  
Enter the number of elements: 5  
Enter 5 integers:  
12  
22  
33  
44  
55  
Sorted array is:  
12 22 33 44 55
```

2) Write a C program to find max-min using divide and Conquer
Program:

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

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

DAY-2 PROGRAMS

```
int max, min;
int a[100];
void maxmin(int i, int j)
{
    int max1, min1, mid;
    if(i==j)
    {
        max = min = a[i];
    }
    else
    {
        if(i == j-1)
        {
            if(a[i] < a[j])
            {
                max = a[j];
                min = a[i];
            }
            else
            {
                max = a[i];
                min = a[j];
            }
        }
        else
        {
            mid = (i+j)/2;
```

DAY-2 PROGRAMS

```
    maxmin(i, mid);
    max1 = max; min1 = min;
    maxmin(mid+1, j);
    if(max < max1)
        max = max1;
    if(min > min1)
        min = min1;
}
}
}

int main ()
{
    int i, num;
    printf ("\nEnter the total number of numbers : ");
    scanf ("%d",&num);
    printf ("Enter the numbers : \n");
    for (i=1;i<=num;i++)
        scanf ("%d",&a[i]);

    max = a[0];
    min = a[0];
    maxmin(1, num);
    printf ("Minimum element in an array : %d\n", min);
    printf ("Maximum element in an array : %d\n", max);
    return 0;
}
```

DAY-2 PROGRAMS

p

```
1 #include<stdio.h>
2 #include<stdio.h>
3 int max, min;
4 int a[100];
5 void maxmin(int i, int j)
6 {
7     int max1, min1, mid;
8     if(i==j)
9     {
10         max = min = a[i];
11     }
12     else
13     {
14         if(i == j-1)
15         {
16             if(a[i] < a[j])
17             {
18                 max = a[j];
19                 min = a[i];
20             }
21             else
22             {
23                 max = a[i];
24                 min = a[j];
25             }
26         }
27     }
28 }
```

```
/tmp/1qCbQwPRRp.o
Enter the total number of numbers : 5
Enter the numbers :
56
66
77
8
8
88
Minimum element in an array : 6
Maximum element in an array : 88
```

3) 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);

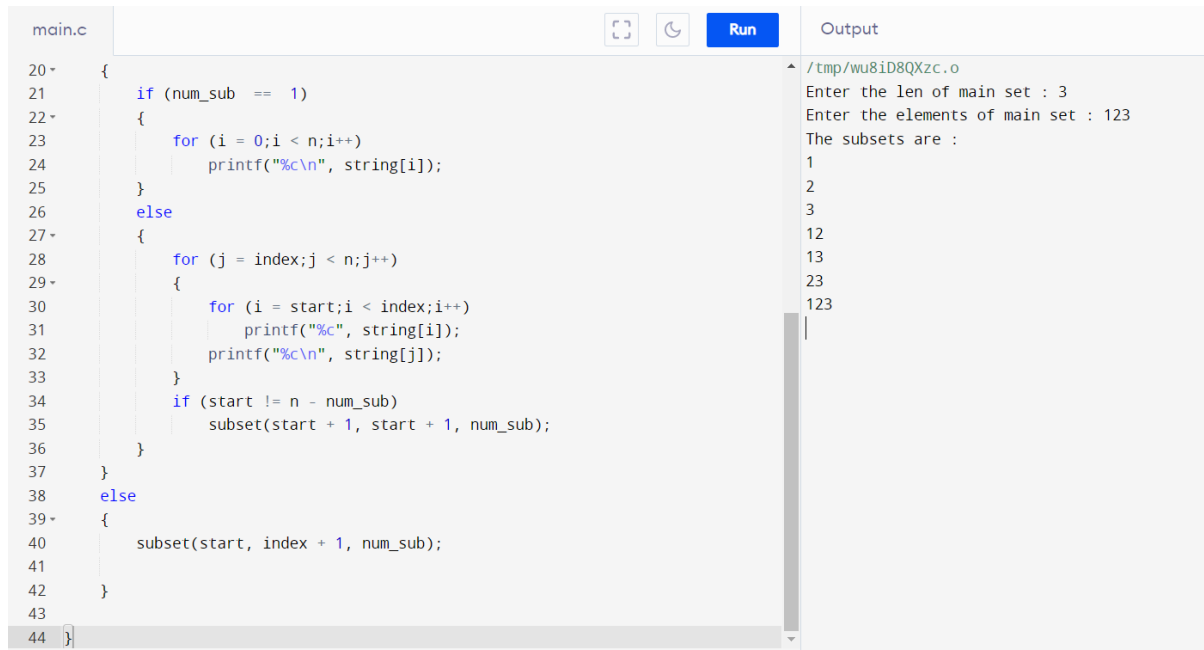
    n = len;
```

DAY-2 PROGRAMS

```
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
```

DAY-2 PROGRAMS

```
{  
    subset(start, index + 1, num_sub);  
  
}  
  
}
```



The screenshot shows a C program in a text editor. The code defines a recursive function `subset` that prints all subsets of a given string. The main function prompts the user for the length of the main set and the elements of the main set. The output shows the subsets of the set {1, 2, 3}.

```
main.c  
20+ {  
21     if (num_sub == 1)  
22+     {  
23         for (i = 0; i < n; i++)  
24             printf("%c\n", string[i]);  
25     }  
26     else  
27+     {  
28         for (j = index; j < n; j++)  
29+         {  
30             for (i = start; i < index; i++)  
31                 printf("%c", string[i]);  
32             printf("%c\n", string[j]);  
33         }  
34         if (start != n - num_sub)  
35             subset(start + 1, start + 1, num_sub);  
36     }  
37 }  
38 else  
39+ {  
40     subset(start, index + 1, num_sub);  
41 }  
42 }  
43  
44 }
```

Output

```
/tmp/wu8iD8QXzc.o  
Enter the len of main set : 3  
Enter the elements of main set : 123  
The subsets are :  
1  
2  
3  
12  
13  
23  
123
```

4) Write a program to compute container loader Problem for the given values and estimate time complexity.

N=8 be total no of containers having weights (w1, w2, w3,...w8) = [50, 100, 30, 80, 90, 200, 150, 20]. Capacity value = 100

Program:

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

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

```
#define MAX_ITEMS 100
```

```
#define MAX_WEIGHT 100
```

```
int weight[MAX_ITEMS];
```


DAY-2 PROGRAMS

```
int value[MAX_ITEMS];  
int dp[MAX_ITEMS][MAX_WEIGHT];
```

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

```
int knapsack(int n, int w) {  
    int i, j;  
    for (i = 0; i <= n; i++) {  
        for (j = 0; j <= w; j++) {  
            if (i == 0 || j == 0) {  
                dp[i][j] = 0;  
            } else if (weight[i-1] <= j) {  
                dp[i][j] = max(value[i-1] + dp[i-1][j-weight[i-1]], dp[i-1][j]);  
            } else {  
                dp[i][j] = dp[i-1][j];  
            }  
        }  
    }  
    return dp[n][w];  
}
```

```
int main() {  
    int n = 4;  
    int w = 10;  
    weight[0] = 1;
```

DAY-2 PROGRAMS

```
weight[1] = 2;
weight[2] = 4;
weight[3] = 5;
value[0] = 5;
value[1] = 4;
value[2] = 6;
value[3] = 8;

int result = knapsack(n, w);

printf("Result: %d\n", result);

return 0;
}
```



The screenshot shows a C++ IDE with a file named 'main.cpp'. The code implements a knapsack problem solution. It defines arrays for 'weight' and 'value' with 4 elements each. The 'weight' array has values [1, 2, 4, 5] and the 'value' array has values [5, 4, 6, 8]. The 'knapsack' function is called with n=4 and w=10. The output of the program is 'Result: 19'.

```
main.cpp
21- } else if (weight[i-1] <= j) {
22-     dp[i][j] = max(value[i-1] + dp[i-1][j-weight[i-1]], dp[i-1][j]);
23- } else {
24-     dp[i][j] = dp[i-1][j];
25- }
26- }
27- }
28- return dp[n][w];
29- }
30-
31- int main() {
32-     int n = 4;
33-     int w = 10;
34-     weight[0] = 1;
35-     weight[1] = 2;
36-     weight[2] = 4;
37-     weight[3] = 5;
38-     value[0] = 5;
39-     value[1] = 4;
40-     value[2] = 6;
41-     value[3] = 8;
42-     int result = knapsack(n, w);
43-     printf("Result: %d\n", result);
44-     return 0;
45- }
```

Output

```
/tmp/RfBWyavj7o.o
Result: 19
```

5) Write a program to find a minimum spanning tree using prims technique for the given graph

Program:

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

```
#include <limits.h>
```

DAY-2 PROGRAMS

```
#define V 5
```

```
int minKey(int key[], bool mstSet[]) {  
    int min = INT_MAX, minIndex;  
    for (int v = 0; v < V; v++)  
        if (mstSet[v] == false && key[v] < min)  
            min = key[v], minIndex = v;  
    return minIndex;  
}
```

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

```
void primMST(int graph[V][V]) {  
    int parent[V];  
    int key[V];  
    bool mstSet[V];  
    for (int i = 0; i < V; i++)  
        key[i] = INT_MAX, mstSet[i] = false;  
    key[0] = 0;  
    parent[0] = -1;  
    for (int count = 0; count < V - 1; count++) {  
        int u = minKey(key, mstSet);
```

DAY-2 PROGRAMS

```
mstSet[u] = true;
for (int v = 0; v < V; v++)
    if (graph[u][v] && mstSet[v] == false && graph[u][v] < key[v])
        parent[v] = u, key[v] = graph[u][v];
}
printMST(parent, graph);
}

int main() {
    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;
}
```

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

```
-----
Process exited after 0.05377 seconds with return value 0
Press any key to continue . . .
```