

```
1 #include<stdio.h>
2 #include<stdlib.h>
3
4 int comparator(const void *p1,const void *p2)//used by qsort()
5 {
6     const int (*x)[3]=p1;
7     const int (*y)[3]=p2;
8
9     return (*x)[2]-(*y)[2];
10 }
11
12 void makeSet(int parent[],int rank[],int n)
13 {
14     for(int i=0;i<n;i++)
15     {
16         parent[i]=i;
17         rank[i]=0;
18     }
19 }
20
21 int findParent(int parent[],int component)
22 {
23     if(parent[component]==component)
24         return component;
25
26     return parent[component]=findParent(parent,parent[component]);
27 }
28
29 void unionSet(int u,int v,int parent[],int rank[],int n)
30 {
31
32     u=findParent(parent,u);
33     v=findParent(parent,v);
34
35     if(rank[u]<rank[v])
36     {
37         parent[u]=v;
38     }
39     else if(rank[u]<rank[v])
40     {
41         parent[v]=u;
42     }
43 }
```

/tmp/Yu71o9aJGN.o

Following are the edges in the constructed MST

2 -- 3 == 4

0 -- 3 == 5

0 -- 1 == 10

Minimum Cost Spanning Tree: 19

```
1 #include <stdio.h>
2
3 #define max 10
4
5 int a[11] = { 10, 14, 19, 26, 27, 31, 33, 35, 42, 44, 0 };
6 int b[10];
7
8 void merging(int low, int mid, int high) {
9     int l1, l2, i;
10
11     for(l1 = low, l2 = mid + 1, i = low; l1 <= mid && l2 <= high; i++) {
12         if(a[l1] <= a[l2])
13             b[i] = a[l1++];
14         else
15             b[i] = a[l2++];
16     }
17
18     while(l1 <= mid)
19         b[i++] = a[l1++];
20
21     while(l2 <= high)
22         b[i++] = a[l2++];
23
24     for(i = low; i <= high; i++)
25         a[i] = b[i];
26 }
27
28 void sort(int low, int high) {
29     int mid;
30
31     if(low < high) {
32         mid = (low + high) / 2;
33         sort(low, mid);
```

/tmp/HYounQONY0.o

List before sorting

10 14 19 26 27 31 33 35 42 44 0

List after sorting

0 10 14 19 26 27 31 33 35 42 44

```
23     max = a[i];
24     min = a[j];
25 }
26 }
27 else
28 {
29     mid = (i+j)/2;
30     maxmin(i, mid);
31     max1 = max; min1 = min;
32     maxmin(mid+1, j);
33     if(max < max1)
34         max = max1;
35     if(min > min1)
36         min = min1;
37 }
38 }
39 }
40 int main ()
41 {
42     int i, num;
43     printf ("\nEnter the total number of numbers : ");
44     scanf ("%d",&num);
45     printf ("Enter the numbers : \n");
46     for (i=1;i<=num;i++)
47         scanf ("%d",&a[i]);
48
49     max = a[0];
50     min = a[0];
51     maxmin(1, num);
52     printf ("Minimum element in an array : %d\n", min);
53     printf ("Maximum element in an array : %d\n", max);
54     return 0;
55 }
```

```
^ /tmp/Vsc57Zkz7z.o
Enter the total number of numbers : 2
Enter the numbers :
8
0
Minimum element in an array : 0
Maximum element in an array : 8
```

```
1 #include<stdio.h>
2
3 void knapsack(int n, float weight[], float profit[], float capacity) {
4     float x[20], tp = 0;
5     int i, j, u;
6     u = capacity;
7
8     for (i = 0; i < n; i++)
9         x[i] = 0.0;
10
11     for (i = 0; i < n; i++) {
12         if (weight[i] > u)
13             break;
14         else {
15             x[i] = 1.0;
16             tp = tp + profit[i];
17             u = u - weight[i];
18         }
19     }
20
21     if (i < n)
22         x[i] = u / weight[i];
23
24     tp = tp + (x[i] * profit[i]);
25
26     printf("\nThe result vector is:- ");
27     for (i = 0; i < n; i++)
28         printf("%f\t", x[i]);
29
30     printf("\nMaximum profit is:- %f", tp);
31
32 }
33
```

/tmp/SZp5z9jUmU.o

Enter the no. of objects:- 2

Enter the wts and profits of each object:- 12

43

3

3

4

6Enter the capacity of knapsack:- 14

The result vector is:- 1.000000 0.916667

Maximum profit is:- 6.750000

```
9  scanf("%d", &end);
10
11  printf("All prime numbers between 1 to %d are:\n", end);
12
13  /* Find all Prime numbers between 1 to end */
14  for(i=2; i<=end; i++)
15  {
16      /* Assume that the current number is Prime */
17      isPrime = 1;
18
19      /* Check if the current number i is prime or not */
20      for(j=2; j<=i/2; j++)
21      {
22          /*
23           * If i is divisible by any number other than 1 and self
24           * then it is not prime number
25           */
26          if(i%j==0)
27          {
28              isPrime = 0;
29              break;
30          }
31      }
32
33      /* If the number is prime then print */
34      if(isPrime==1)
35      {
36          printf("%d, ", i);
37      }
38  }
39
40  return 0;
41 }
```

/tmp/H4QRBv5SQB.o

Find prime numbers between 1 to : 50

All prime numbers between 1 to 50 are:

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47,