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++;
    }
```

```
    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++)
```

```
 {
     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

```
        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++) {
        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;
}
```

Output

```
/tmp/wu8iD8QXzc.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>
```

```
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;
```

```
   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;
}
```

p

```c
#include<stdio.h>
#include<stdio.h>
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];
   }
```

```
/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

```c
#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);

    }

}
```

main.c

```c
    {
        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

```
/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];
```

```
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;
```

```
  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;
}
```

main.c

Run

```
      } 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;
  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;
}
```

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>
```

```
#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);
```

```
        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 . . .
```