1./* Insertion sort ascending order */

```
#include <stdio.h>

int main()
{
 int n, array[1000], c, d, t;

 printf("Enter number of elements\n");
 scanf("%d", &n);

 printf("Enter %d integers\n", n);

 for (c = 0; c < n; c++) {
   scanf("%d", &array[c]);
 }

 for (c = 1 ; c <= n - 1; c++) {
   d = c;

   while ( d > 0 && array[d] < array[d-1]) {
     t          = array[d];
     array[d]   = array[d-1];
     array[d-1] = t;

     d--;
   }
 }

 printf("Sorted list in ascending order:\n");

 for (c = 0; c <= n - 1; c++) {
   printf("%d\n", array[c]);
 }

 return 0;
```

```
        }
```

2.// Selection sort in C

```
#include <stdio.h>

// function to swap the the position of two elements
void swap(int *a, int *b) {
  int temp = *a;
  *a = *b;
  *b = temp;
}

void selectionSort(int array[], int size) {
  for (int step = 0; step < size - 1; step++) {
    int min_idx = step;
    for (int i = step + 1; i < size; i++) {

      // To sort in descending order, change > to < in this line.
      // Select the minimum element in each loop.
      if (array[i] < array[min_idx])
        min_idx = i;
    }

    // put min at the correct position
    swap(&array[min_idx], &array[step]);
  }
}

// function to print an array
void printArray(int array[], int size) {
  for (int i = 0; i < size; ++i) {
    printf("%d  ", array[i]);
  }
  printf("\n");
}

// driver code
```

```
int main() {
  int data[] = {20, 12, 10, 15, 2};
  int size = sizeof(data) / sizeof(data[0]);
  selectionSort(data, size);
  printf("Sorted array in Acsending Order:\n");
  printArray(data, size);
}
```

3./* Bubble sort code */

```
#include <stdio.h>

int main()
{
  int array[100], n, c, d, swap;

  printf("Enter number of elements\n");
  scanf("%d", &n);

  printf("Enter %d integers\n", n);

  for (c = 0; c < n; c++)
    scanf("%d", &array[c]);

  for (c = 0 ; c < n - 1; c++)
  {
    for (d = 0 ; d < n - c - 1; d++)
    {
      if (array[d] > array[d+1]) /* For decreasing order use < */
      {
        swap       = array[d];
        array[d]   = array[d+1];
        array[d+1] = swap;
      }
    }
  }
```

```
  printf("Sorted list in ascending order:\n");

  for (c = 0; c < n; c++)
     printf("%d\n", array[c]);

  return 0;
}
```

4.Merge sort - Code

```
#include <stdio.h>

// function to sort the subsection a[i .. j] of the array a[]
void merge_sort(int i, int j, int a[], int aux[]) {
    if (j <= i) {
        return;     // the subsection is empty or a single element
    }
    int mid = (i + j) / 2;

    // left sub-array is a[i .. mid]
    // right sub-array is a[mid + 1 .. j]

    merge_sort(i, mid, a, aux);     // sort the left sub-array recursively
    merge_sort(mid + 1, j, a, aux);     // sort the right sub-array recursively

    int pointer_left = i;       // pointer_left points to the beginning of the left sub-array
    int pointer_right = mid + 1;        // pointer_right points to the beginning of the right
sub-array
    int k;      // k is the loop counter

    // we loop from i to j to fill each element of the final merged array
    for (k = i; k <= j; k++) {
```

```
        if (pointer_left == mid + 1) {      // left pointer has reached the limit
            aux[k] = a[pointer_right];
            pointer_right++;
        } else if (pointer_right == j + 1) {        // right pointer has reached the limit
            aux[k] = a[pointer_left];
            pointer_left++;
        } else if (a[pointer_left] < a[pointer_right]) {        // pointer left points to smaller
element
            aux[k] = a[pointer_left];
            pointer_left++;
        } else {        // pointer right points to smaller element
            aux[k] = a[pointer_right];
            pointer_right++;
        }
    }

    for (k = i; k <= j; k++) {      // copy the elements from aux[] to a[]
        a[k] = aux[k];
    }
}


int main() {
  int a[100], aux[100], n, i, d, swap;

  printf("Enter number of elements in the array:\n");
  scanf("%d", &n);

  printf("Enter %d integers\n", n);

  for (i = 0; i < n; i++)
    scanf("%d", &a[i]);

  merge_sort(0, n - 1, a, aux);

  printf("Printing the sorted array:\n");

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

```
  return 0;
}


5.#include<stdio.h>

void create(int []);
void down_adjust(int [],int);

void main()
{
        int heap[30],n,i,last,temp;
        printf("Enter no. of elements:");
        scanf("%d",&n);
        printf("\nEnter elements:");
        for(i=1;i<=n;i++)
                scanf("%d",&heap[i]);

        //create a heap
        heap[0]=n;
        create(heap);

        //sorting
        while(heap[0] > 1)
        {
                //swap heap[1] and heap[last]
                last=heap[0];
                temp=heap[1];
                heap[1]=heap[last];
                heap[last]=temp;
                heap[0]--;
                down_adjust(heap,1);
        }

        //print sorted data
        printf("\nArray after sorting:\n");
        for(i=1;i<=n;i++)
                printf("%d ",heap[i]);
```

```
}

void create(int heap[])
{
        int i,n;
        n=heap[0]; //no. of elements
        for(i=n/2;i>=1;i--)
                down_adjust(heap,i);
}

void down_adjust(int heap[],int i)
{
        int j,temp,n,flag=1;
        n=heap[0];

        while(2*i<=n && flag==1)
        {
                j=2*i;   //j points to left child
                if(j+1<=n && heap[j+1] > heap[j])
                        j=j+1;
                if(heap[i] > heap[j])
                        flag=0;
                else
                {
                        temp=heap[i];
                        heap[i]=heap[j];
                        heap[j]=temp;
                        i=j;
                }
        }
}
```