```c
#include <stdio.h>

int main()
{
  int n, array[10], c, d, t, flag = 0;

  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++) {
    t = array[c];

    for (d = c - 1 ; d >= 0; d--) {
      if (array[d] > t) {
        array[d+1] = array[d];
        flag = 1;
      }
      else
        break;
    }
    if (flag)
      array[d+1] = t;
  }
```

2.

```c
#include <stdio.h>
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++) {
```

```
      if (array[i] < array[min_idx])
        min_idx = i;
    }


    swap(&array[min_idx], &array[step]);
  }
}


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


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

3.

```
#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>
void merge_sort(int i, int j, int a[], int aux[]) {
    if (j <= i) {
        return;
    }
    int mid = (i + j) / 2;

    merge_sort(i, mid, a, aux);
    merge_sort(mid + 1, j, a, aux);
    int pointer_left = i;
    int pointer_right = mid + 1;
    int k;      // k is the loop counter


    for (k = i; k <= j; k++) {
        if (pointer_left == mid + 1) {
            aux[k] = a[pointer_right];
            pointer_right++;
```

```
        } else if (pointer_right == j + 1) {
            aux[k] = a[pointer_left];
            pointer_left++;
        } else if (a[pointer_left] < a[pointer_right]) {
            aux[k] = a[pointer_left];
            pointer_left++;
        } else {
            aux[k] = a[pointer_right];
            pointer_right++;
        }
    }

    for (k = i; k <= j; k++) {
        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;
                }
        }
}
```