```c
// Q. WAP on: -

//  Bubble sort
//  Selection sort
//  Insertion sort
//  Heap sort

// For each sorting problem given, find out the time complexity for Best case and worst case.
#include <stdio.h>
#define MAX 100
void ascenb(int a[], int n);
void descenb(int a[], int n);
void ascens(int a[], int n);
void descens(int a[], int n);
void asceni(int a[], int n);
void desceni(int a[], int n);
void ascenh(int a[], int n, int i);
void descenh(int a[], int n, int i);
void heapSort(int a[], int n);
void heapSort2(int a[], int n);
void printArray(int a[], int n);

int main()
{
    int a[MAX], n;
    int i, j, temp, choice, choi;
    printf("Enter total number of elements: ");
    scanf("%d", &n);
    printf("Enter array elements: \n");
    for (i = 0; i < n; i++)
    {
        printf("Enter element %d: ", i + 1);
        scanf("%d", &a[i]);
    }
    printf("\n1. Bubble Sort\n2. Selection Sort\n3. Insertion Sort\n4. Heap sort.\n ");
    printf("\nEnter your Choice : ");
    scanf("%d", &choi);

    switch (choi)
    {
    case 1:
        printf("\n1. Ascending Sort\n2. Descending Sort\n");
        printf("\nEnter your Choice : ");
        scanf("%d", &choice);

        switch (choice)
        {
        case 1:
            ascenb(a, n);
            printf("Time complexity:\nBest case: O(n) \nWorst Case: O(n2)");
```

```
            break;
        case 2:
            descenb(a, n);
            printf("Time complexity:\nBest case: O(n) \nWorst Case: O(n2)");
            break;
        }
        break;
    case 2:
        printf("\n1. Ascending Sort\n2. Descending Sort\n");
        printf("\nEnter your Choice : ");
        scanf("%d", &choice);

        switch (choice)
        {
        case 1:
            ascens(a, n);
            printf("Time complexity:\nBest case: O(n2) \nWorst Case: O(n2)");
            break;
        case 2:
            descens(a, n);
            printf("Time complexity:\nBest case: O(n2) \nWorst Case: O(n2)");
            break;
        }
        break;
    case 3:
        printf("\n1. Ascending Sort\n2. Descending Sort\n");
        printf("\nEnter your Choice : ");
        scanf("%d", &choice);

        switch (choice)
        {
        case 1:
            asceni(a, n);
            printf("Time complexity:\nBest case: O(n) \nWorst Case: O(n2)");
            break;
        case 2:
            desceni(a, n);
            printf("Time complexity:\nBest case: O(n) \nWorst Case: O(n2)");
            break;
        }
        break;
    case 4:
        printf("\n1. Ascending Sort\n2. Descending Sort\n");
        printf("\nEnter your Choice : ");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            heapSort(a, n);
            printf(" Sorted array is \n");
```

```
            printArray(a, n);
            printf("Time complexity:\nBest case: O(nlogn) \nWorst Case: O(nlogn)");
            break;
        case 2:
            heapSort2(a, n);
            printf(" Sorted array is \n");
            printArray(a, n);
            printf("Time complexity:\nBest case: O(nlogn) \nWorst Case: O(nlogn)");
            break;
        }
    }
}
void ascenb(int a[], int n)
{
    int i, j, temp;
    for (i = 0; i < (n - 1); i++)
    {
        for (j = 0; j < (n - i - 1); j++)
        {
            if (a[j] > a[j + 1])
            {
                temp = a[j];
                a[j] = a[j + 1];
                a[j + 1] = temp;
            }
        }
    }
    printf("Array elements in Ascending Order:\n");
    for (i = 0; i < n; i++)
        printf("%d ", a[i]);
    printf("\n");
}
void descenb(int a[], int n)
{
    int i, j, temp;
    for (i = 0; i < (n - 1); i++)
    {
        for (j = 0; j < (n - i - 1); j++)
        {
            if (a[j] > a[j + 1])
            {
                temp = a[j];
                a[j] = a[j + 1];
                a[j + 1] = temp;
            }
        }
    }
    for (i = 0; i < (n - 1); i++)
    {
        for (j = 0; j < (n - i - 1); j++)
```

```
        {
            if (a[j] < a[j + 1])
            {
                temp = a[j];
                a[j] = a[j + 1];
                a[j + 1] = temp;
            }
        }
    }
    printf("\Array elements in Descending Order:\n");
    for (i = 0; i < n; i++)
        printf("%d ", a[i]);
    printf("\n");
}
void ascens(int a[], int n)
{
    int i, j, temp, pos;
    for (i = 0; i < (n); i++)
    {
        pos = i;
        for (j = i + 1; j < n; j++)
        {
            if (a[pos] > a[j])
            {
                pos = j;
            }
            if (pos != i)
            {
                temp = a[i];
                a[i] = a[pos];
                a[pos] = temp;
            }
        }
    }
    printf("\Array elements in \Ascending Order:\n");
    for (i = 0; i < n; i++)
        printf("%d ", a[i]);
    printf("\n");
}
void descens(int a[], int n)
{
    int i, j, temp, pos;
    for (i = 0; i < (n); i++)
    {
        pos = i;
        for (j = i + 1; j < n; j++)
        {
            if (a[pos] > a[j])
            {
                pos = j;
```

```
            }
            if (pos != i)
            {
                temp = a[i];
                a[i] = a[pos];
                a[pos] = temp;
            }
        }
    }
    for (i = 0; i < (n); i++)
    {
        pos = i;
        for (j = i + 1; j < n; j++)
        {
            if (a[pos] < a[j])
            {
                pos = j;
            }
            if (pos != i)
            {
                temp = a[i];
                a[i] = a[pos];
                a[pos] = temp;
            }
        }
    }
    printf("\nArray elements in Descending Order:\n");
    for (i = 0; i < n; i++)
        printf("%d ", a[i]);
    printf("\n");
}
void asceni(int a[], int n)
{
    int i, j, temp;
    for (i = 1; i < (n); i++)
    {
        j = i;
        while (j > 0 && a[j] < a[j - 1])
        {
            temp = a[j];
            a[j] = a[j - 1];
            a[j - 1] = temp;
            j--;
        }
    }
    printf("\nArray elements in Ascending Order:\n");
    for (i = 0; i < n; i++)
        printf("%d ", a[i]);
    printf("\n");
}
```

```
void desceni(int a[], int n)
{
    int i, j, temp;
    for (i = 1; i < (n); i++)
    {
        j = i;
        while (j > 0 && a[j] < a[j - 1])
        {
            temp = a[j];
            a[j] = a[j - 1];
            a[j - 1] = temp;
            j--;
        }
    }
    for (i = 1; i < (n); i++)
    {
        j = i;
        while (j > 0 && a[j] > a[j - 1])
        {
            temp = a[j];
            a[j] = a[j - 1];
            a[j - 1] = temp;
            j--;
        }
    }
    printf("\Array elements in Descending Order:\n");
    for (i = 0; i < n; i++)
        printf("%d ", a[i]);
    printf("\n");
}
void ascenh(int a[], int n, int i)
{
    int lr = i, temp;
    int l = 2 * i + 1;
    int r = 2 * i + 2;
    if (l < n && a[l] > a[lr])
        lr = l;

    if (r < n && a[r] > a[lr])
        lr = r;

    if (lr != i)
    {
        temp = a[i];
        a[i] = a[lr];
        a[lr] = temp;
        ascenh(a, n, lr);
    }
}
```

```
void descenh(int a[], int n, int i)
{
    int sm = i, temp;
    int l = 2 * i + 1;
    int r = 2 * i + 2;
    if (l < n && a[l] < a[sm])
        sm = l;
    if (r < n && a[r] < a[sm])
        sm = r;
    if (sm != i)
    {
        temp = a[i];
        a[i] = a[sm];
        a[sm] = temp;
        descenh(a, n, sm);
    }
}

void heapSort(int a[], int n)
{
    int temp;
    for (int i = n / 2 - 1; i >= 0; i--)
        ascenh(a, n, i);
    for (int i = n - 1; i > 0; i--)
    {
        temp = a[0];
        a[0] = a[i];
        a[i] = temp;
        ascenh(a, i, 0);
    }
}
void heapSort2(int a[], int n)
{
    int temp;
    for (int i = n / 2 - 1; i >= 0; i--)
        descenh(a, n, i);
    for (int i = n - 1; i > 0; i--)
    {
        temp = a[0];
        a[0] = a[i];
        a[i] = temp;
        descenh(a, i, 0);
    }
}
void printArray(int a[], int n)
{
    for (int i = 0; i < n; ++i)
        printf("%d ", a[i]);
    printf("\n");
}
```

OUTPUT

- Bubble ascending

```
Enter total number of elements: 5
Enter array elements:
Enter element 1: 3
Enter element 2: 4
Enter element 3: 2
Enter element 4: 5
Enter element 5: 1

1. Bubble Sort
2. Selection Sort
3. Insertion Sort
4. Heap sort.

Enter your Choice : 1

1. Ascending Sort
2. Descending Sort

Enter your Choice : 1
Array elements in Ascending Order:
1 2 3 4 5
Time complexity:
Best case: O(n)
Worst Case: O(n2)
```

- Bubble descending

```
Enter total number of elements: 5
Enter array elements:
Enter element 1: 3
Enter element 2: 4
Enter element 3: 2
Enter element 4: 5
Enter element 5: 1

1. Bubble Sort
2. Selection Sort
3. Insertion Sort
4. Heap sort.

Enter your Choice : 1

1. Ascending Sort
2. Descending Sort

Enter your Choice : 2
Array elements in Descending Order:
5 4 3 2 1
Time complexity:
Best case: O(n)
Worst Case: O(n2)
```

- Selection Ascending

```
Enter total number of elements: 5
```

Enter array elements:
Enter element 1: 3
Enter element 2: 4
Enter element 3: 2
Enter element 4: 5
Enter element 5: 1

1. Bubble Sort
2. Selection Sort
3. Insertion Sort
4. Heap sort.

Enter your Choice : 2

1. Ascending Sort
2. Descending Sort

Enter your Choice : 1
Array elements in Ascending Order:
1 2 3 4 5
Time complexity:
Best case: O(n2)
Worst Case: O(n2)

- Selection descending

Enter total number of elements: 5
Enter array elements:
Enter element 1: 3
Enter element 2: 4
Enter element 3:
2
Enter element 4: 5
Enter element 5: 1

1. Bubble Sort
2. Selection Sort
3. Insertion Sort
4. Heap sort.

Enter your Choice : 2

1. Ascending Sort
2. Descending Sort

Enter your Choice : 2
Array elements in Descending Order:
5 4 3 2 1
Time complexity:
Best case: O(n2)
Worst Case: O(n2)

- Insertion Ascending

Enter total number of elements: 5
Enter array elements:
Enter element 1: 3
Enter element 2: 4

Enter element 3: 2
Enter element 4: 5
Enter element 5: 1

1. Bubble Sort
2. Selection Sort
3. Insertion Sort
4. Heap sort.

Enter your Choice : 3

1. Ascending Sort
2. Descending Sort

Enter your Choice : 1
Array elements in Ascending Order:
1 2 3 4 5
Time complexity:
Best case: O(n)
Worst Case: O(n2)

- Insertion Descending

Enter total number of elements: 5
Enter array elements:
Enter element 1: 3
Enter element 2: 4
Enter element 3: 2
Enter element 4: 5
Enter element 5: 1

1. Bubble Sort
2. Selection Sort
3. Insertion Sort
4. Heap sort.

Enter your Choice : 3

1. Ascending Sort
2. Descending Sort

Enter your Choice : 2
Array elements in Descending Order:
5 4 3 2 1
Time complexity:
Best case: O(n)
Worst Case: O(n2)

- Heap Ascending

Enter total number of elements: 5
Enter array elements:
Enter element 1: 3
Enter element 2: 4
Enter element 3: 2
Enter element 4: 5
Enter element 5: 1

1. Bubble Sort
2. Selection Sort
3. Insertion Sort
4. Heap sort.

Enter your Choice : 4

1. Ascending Sort
2. Descending Sort

Enter your Choice : 1
Sorted array is
1 2 3 4 5
Time complexity:
Best case: O(nlogn)
Worst Case: O(nlogn)

- Heap Descending

Enter total number of elements: 5
Enter array elements:
Enter element 1: 3
Enter element 2: 4
Enter element 3: 2
Enter element 4: 5
Enter element 5: 1

1. Bubble Sort
2. Selection Sort
3. Insertion Sort
4. Heap sort.

Enter your Choice : 4

1. Ascending Sort
2. Descending Sort

Enter your Choice : 2
Sorted array is
5 4 3 2 1
Time complexity:
Best case: O(nlogn)
Worst Case: O(nlogn)