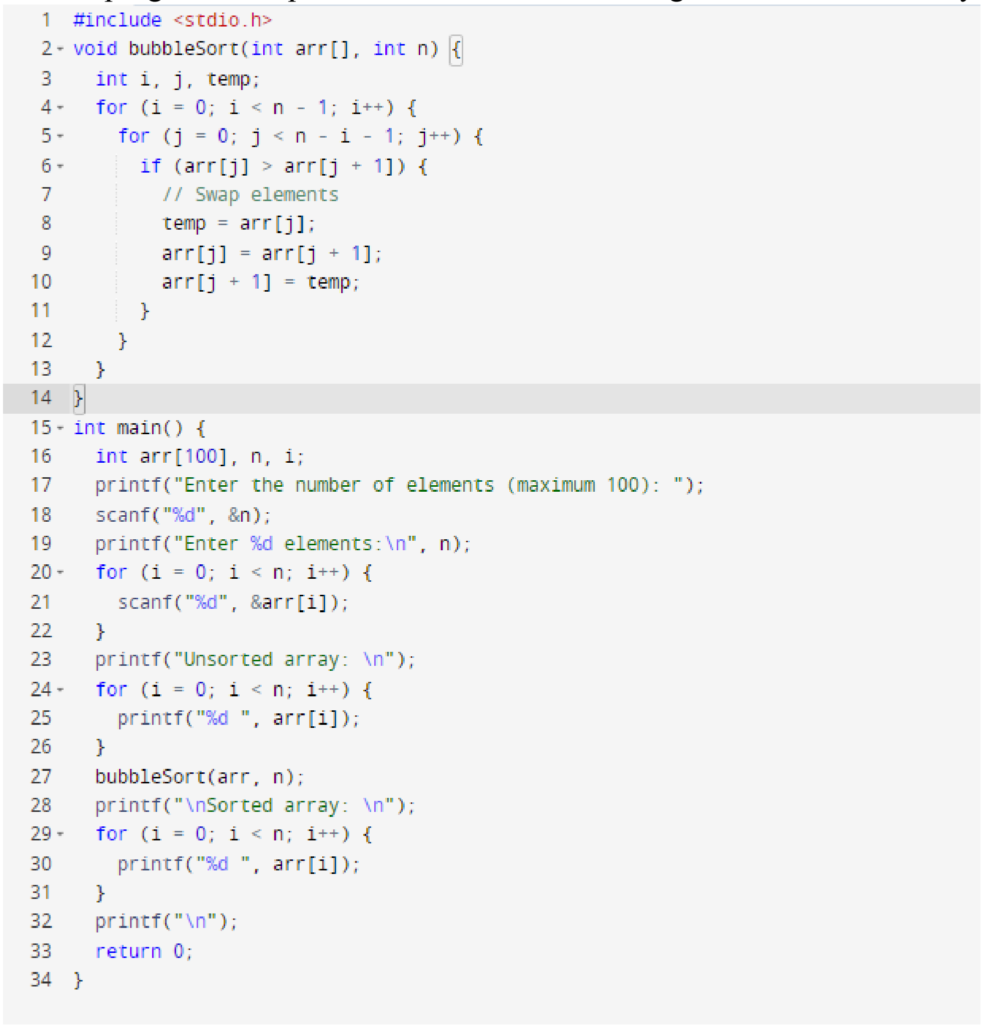
BHAVLEEN KAUR
102155010
3EIC-2

LAB ASSIGNMENT 7
Sorting algorithms

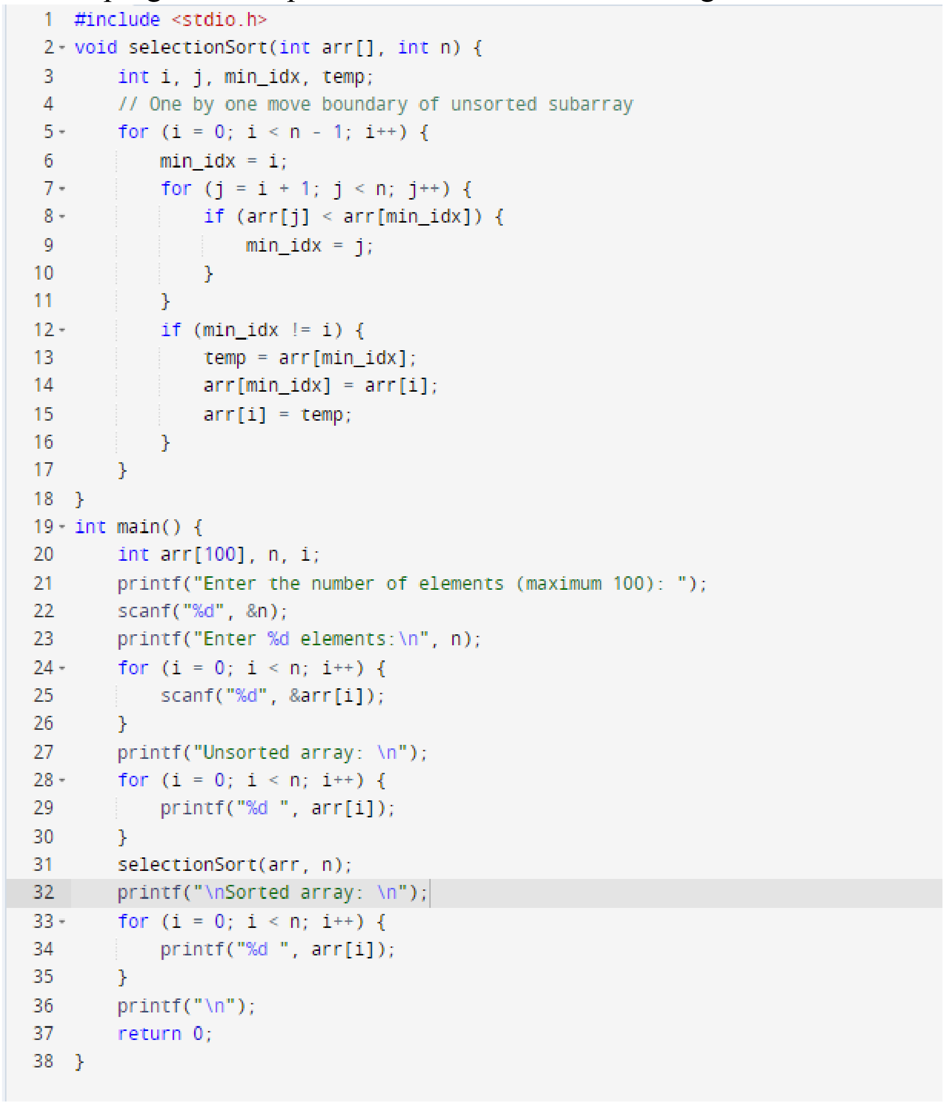
1. Write a program to implement bubble sort for sorting n elements in an array.



```c
#include <stdio.h>
void bubbleSort(int arr[], int n) {
  int i, j, temp;
  for (i = 0; i < n - 1; i++) {
    for (j = 0; j < n - i - 1; j++) {
      if (arr[j] > arr[j + 1]) {
        // Swap elements
        temp = arr[j];
        arr[j] = arr[j + 1];
        arr[j + 1] = temp;
      }
    }
  }
}
int main() {
  int arr[100], n, i;
  printf("Enter the number of elements (maximum 100): ");
  scanf("%d", &n);
  printf("Enter %d elements:\n", n);
  for (i = 0; i < n; i++) {
    scanf("%d", &arr[i]);
  }
  printf("Unsorted array: \n");
  for (i = 0; i < n; i++) {
    printf("%d ", arr[i]);
  }
  bubbleSort(arr, n);
  printf("\nSorted array: \n");
  for (i = 0; i < n; i++) {
    printf("%d ", arr[i]);
  }
  printf("\n");
  return 0;
}
```

BHAVLEEN KAUR
102155010
3EIC-2

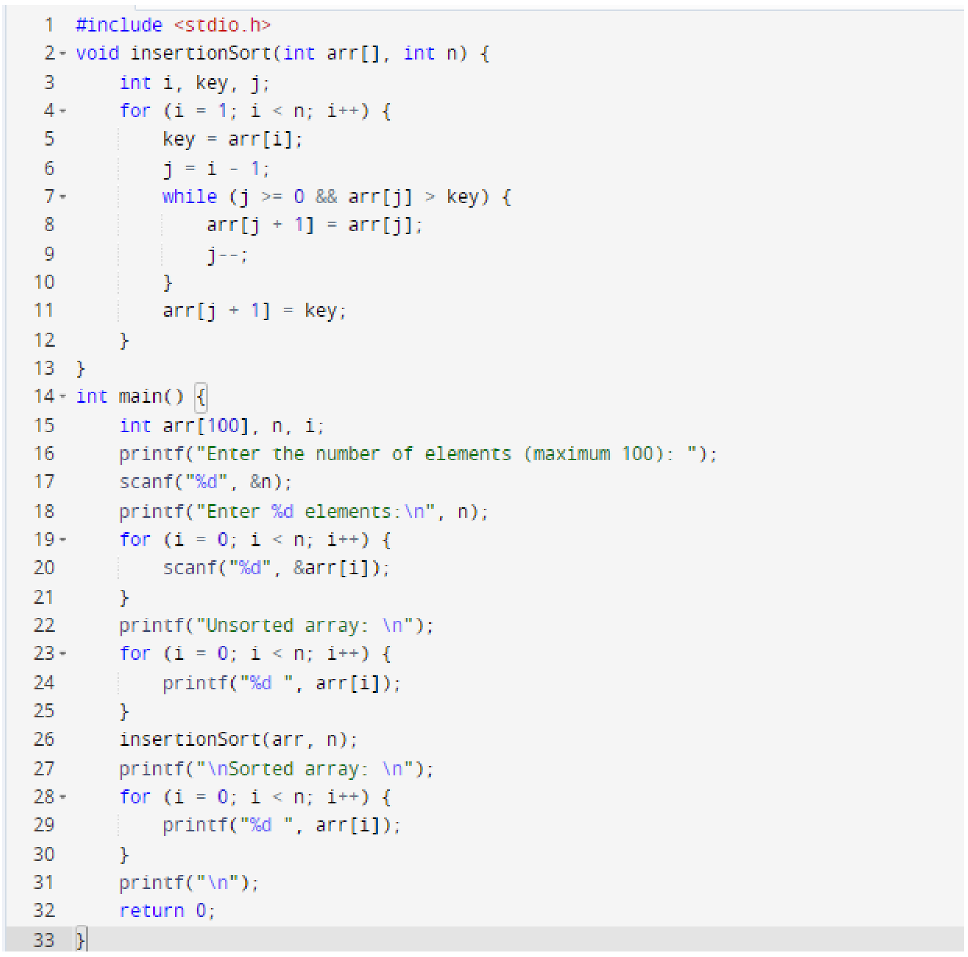
2. Write a program to implement Selection sort for sorting n elements in an array.



```c
#include <stdio.h>
void selectionSort(int arr[], int n) {
    int i, j, min_idx, temp;
    // One by one move boundary of unsorted subarray
    for (i = 0; i < n - 1; i++) {
        min_idx = i;
        for (j = i + 1; j < n; j++) {
            if (arr[j] < arr[min_idx]) {
                min_idx = j;
            }
        }
        if (min_idx != i) {
            temp = arr[min_idx];
            arr[min_idx] = arr[i];
            arr[i] = temp;
        }
    }
}
int main() {
    int arr[100], n, i;
    printf("Enter the number of elements (maximum 100): ");
    scanf("%d", &n);
    printf("Enter %d elements:\n", n);
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    printf("Unsorted array: \n");
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    selectionSort(arr, n);
    printf("\nSorted array: \n");
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
    return 0;
}
```

3. Write a program to implement Insertion sort for sorting n elements in an array.\

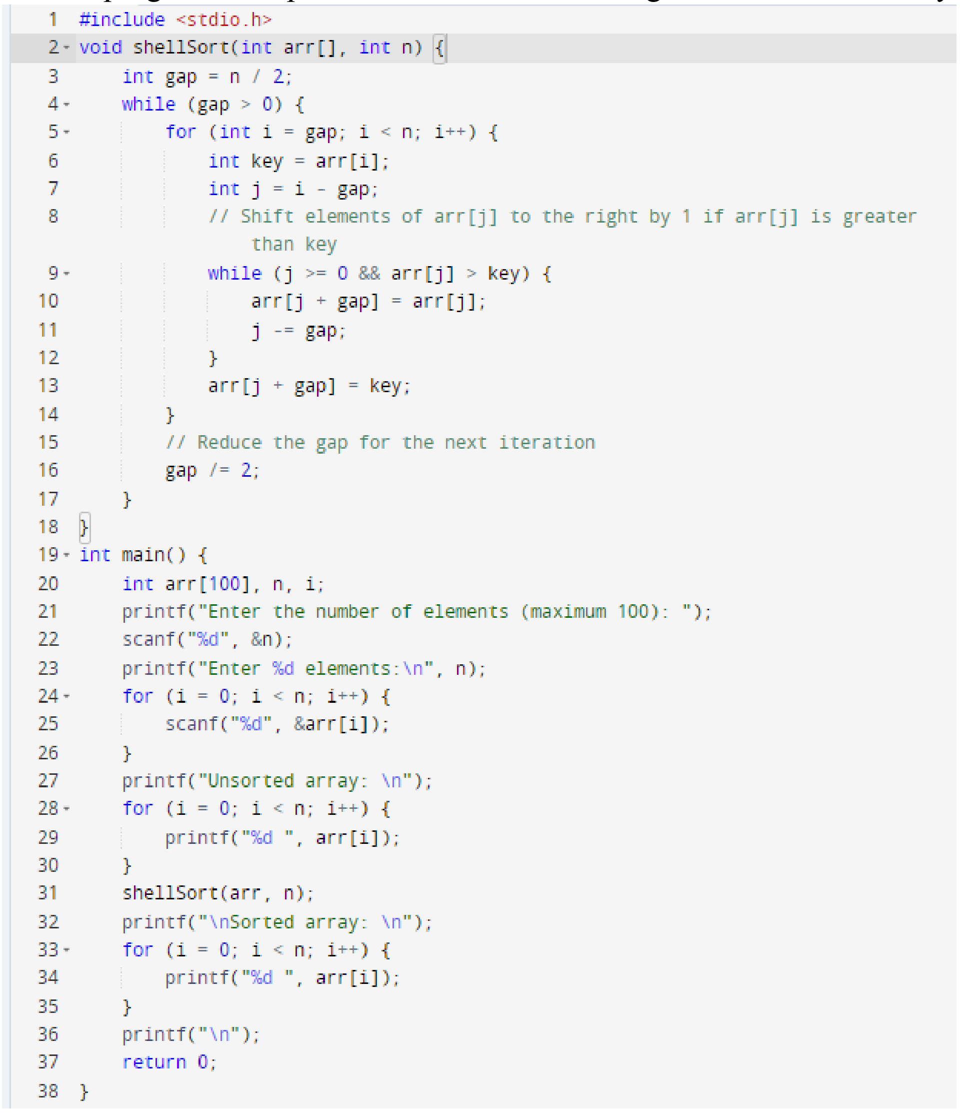
BHAVLEEN KAUR
102155010
3EIC-2



```c
#include <stdio.h>
void insertionSort(int arr[], int n) {
    int i, key, j;
    for (i = 1; i < n; i++) {
        key = arr[i];
        j = i - 1;
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = key;
    }
}
int main() {
    int arr[100], n, i;
    printf("Enter the number of elements (maximum 100): ");
    scanf("%d", &n);
    printf("Enter %d elements:\n", n);
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    printf("Unsorted array: \n");
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    insertionSort(arr, n);
    printf("\nSorted array: \n");
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
    return 0;
}
```

BHAVLEEN KAUR
102155010
3EIC-2

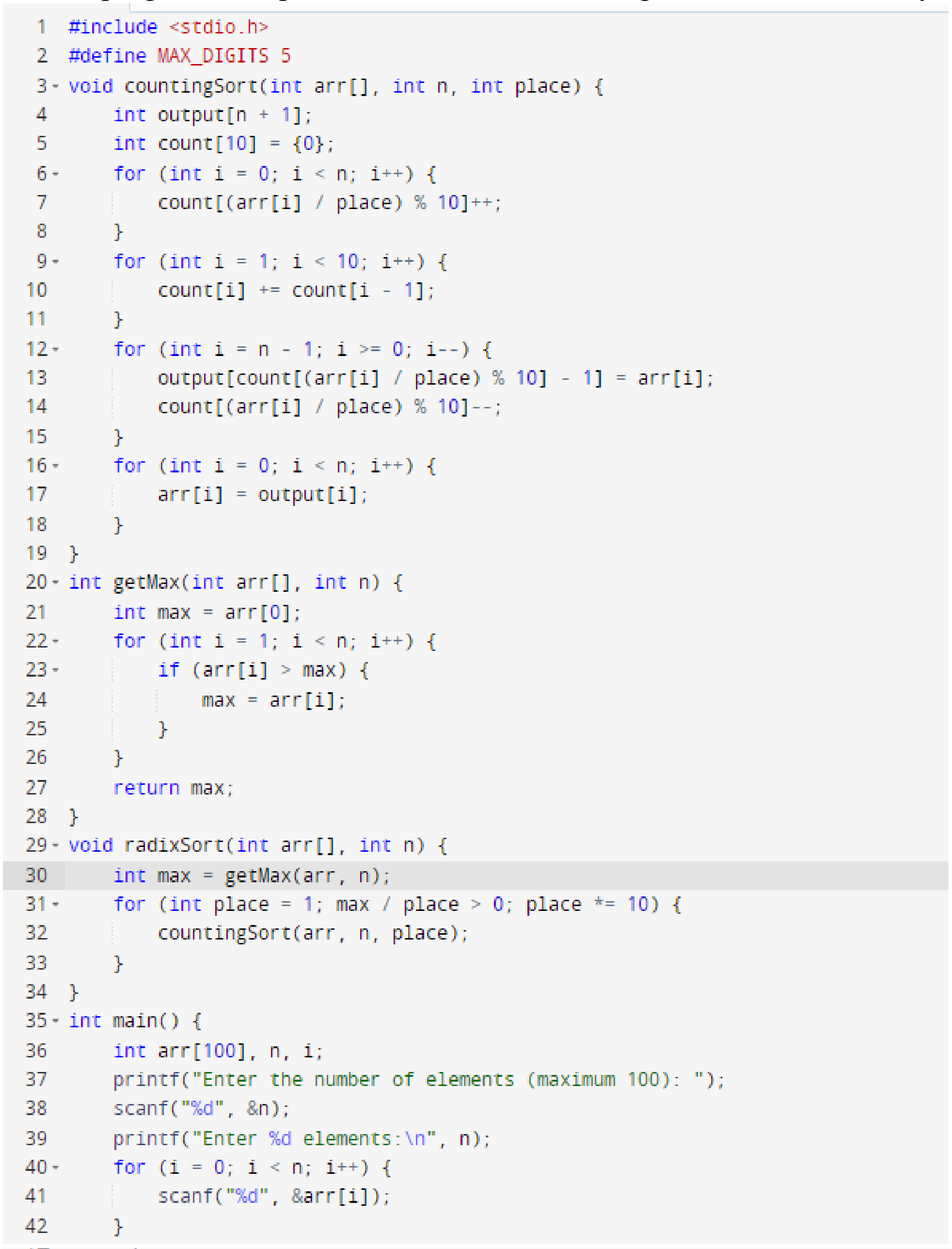
4. Write a program to implement Shell sort for sorting n elements in an array.



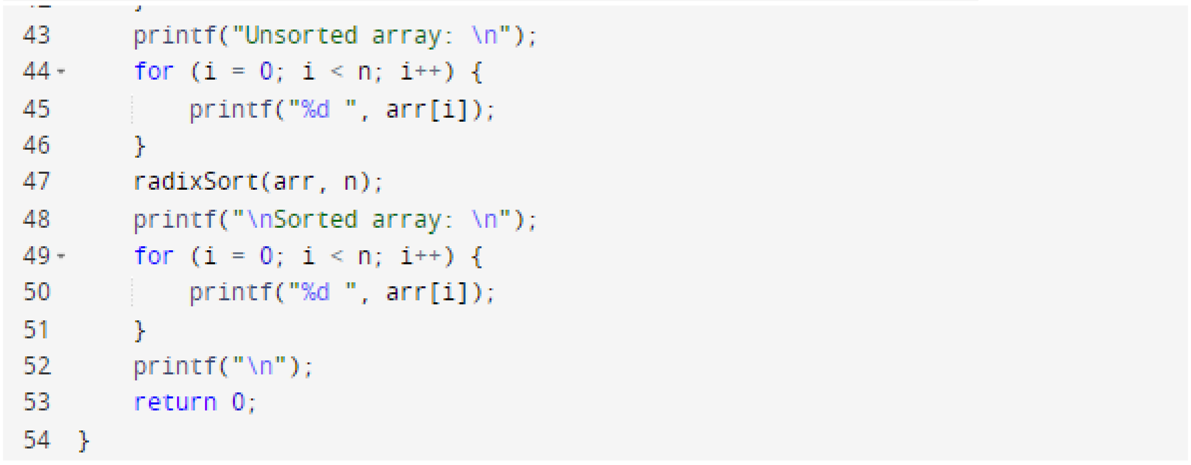
```c
#include <stdio.h>
void shellSort(int arr[], int n) {
    int gap = n / 2;
    while (gap > 0) {
        for (int i = gap; i < n; i++) {
            int key = arr[i];
            int j = i - gap;
            // Shift elements of arr[j] to the right by 1 if arr[j] is greater than key
            while (j >= 0 && arr[j] > key) {
                arr[j + gap] = arr[j];
                j -= gap;
            }
            arr[j + gap] = key;
        }
        // Reduce the gap for the next iteration
        gap /= 2;
    }
}
int main() {
    int arr[100], n, i;
    printf("Enter the number of elements (maximum 100): ");
    scanf("%d", &n);
    printf("Enter %d elements:\n", n);
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    printf("Unsorted array: \n");
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    shellSort(arr, n);
    printf("\nSorted array: \n");
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
    return 0;
}
```

BHAVLEEN KAUR
102155010
3EIC-2

5. Write a program to implement Radix sort for sorting n elements in an array.



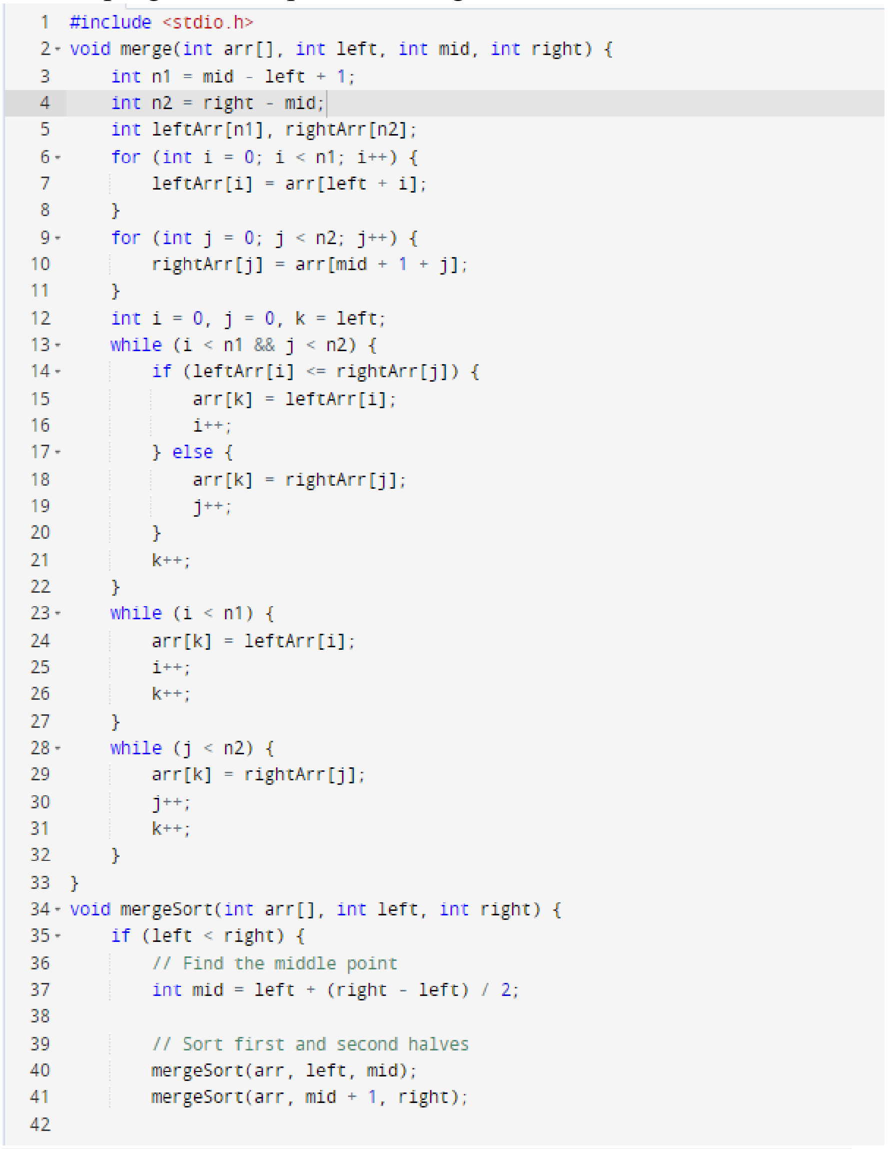
```c
#include <stdio.h>
#define MAX_DIGITS 5
void countingSort(int arr[], int n, int place) {
    int output[n + 1];
    int count[10] = {0};
    for (int i = 0; i < n; i++) {
        count[(arr[i] / place) % 10]++;
    }
    for (int i = 1; i < 10; i++) {
        count[i] += count[i - 1];
    }
    for (int i = n - 1; i >= 0; i--) {
        output[count[(arr[i] / place) % 10] - 1] = arr[i];
        count[(arr[i] / place) % 10]--;
    }
    for (int i = 0; i < n; i++) {
        arr[i] = output[i];
    }
}
int getMax(int arr[], int n) {
    int max = arr[0];
    for (int i = 1; i < n; i++) {
        if (arr[i] > max) {
            max = arr[i];
        }
    }
    return max;
}
void radixSort(int arr[], int n) {
    int max = getMax(arr, n);
    for (int place = 1; max / place > 0; place *= 10) {
        countingSort(arr, n, place);
    }
}
int main() {
    int arr[100], n, i;
    printf("Enter the number of elements (maximum 100): ");
    scanf("%d", &n);
    printf("Enter %d elements:\n", n);
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
```



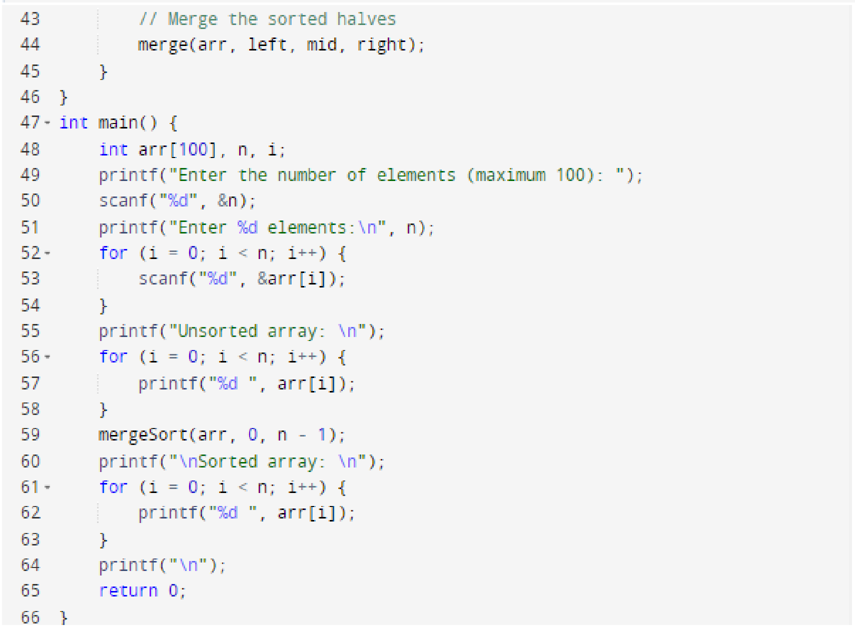
```c
    printf("Unsorted array: \n");
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    radixSort(arr, n);
    printf("\nSorted array: \n");
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
    return 0;
}
```

BHAVLEEN KAUR
102155010
3EIC-2

6. Write a program to implement Merge sort.



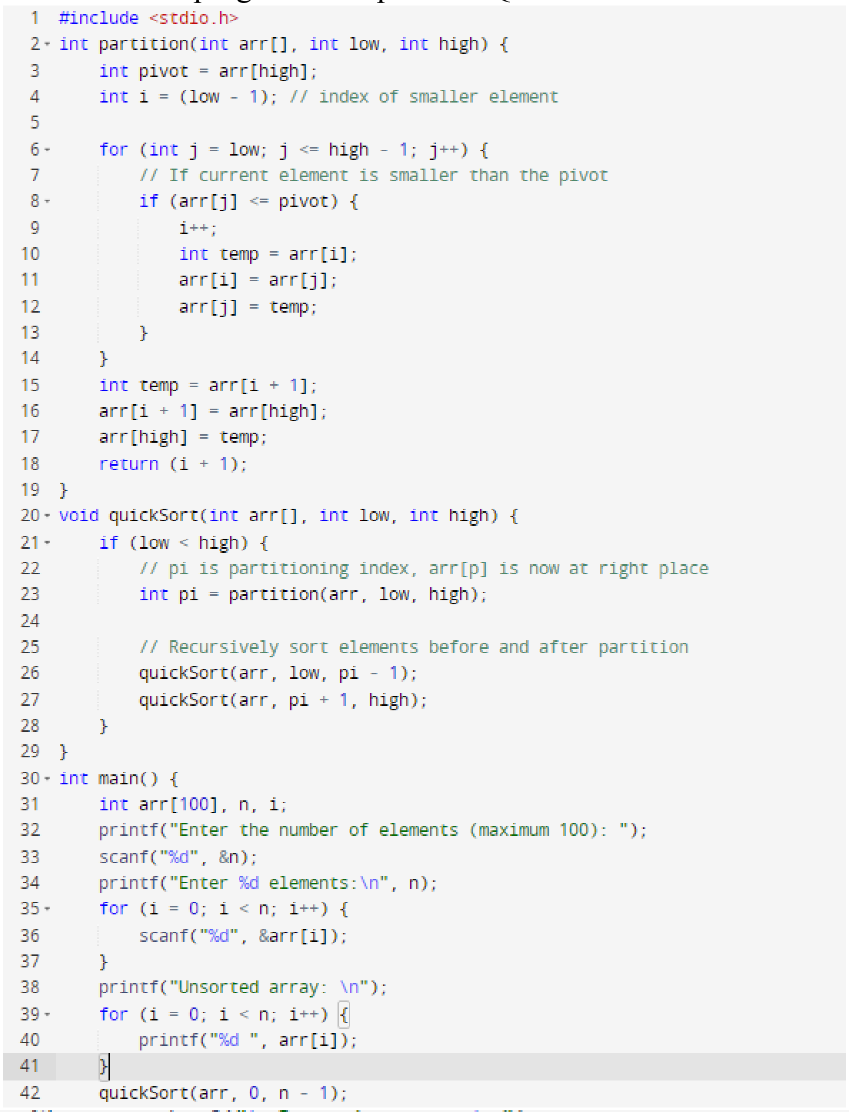
```c
#include <stdio.h>
void merge(int arr[], int left, int mid, int right) {
    int n1 = mid - left + 1;
    int n2 = right - mid;
    int leftArr[n1], rightArr[n2];
    for (int i = 0; i < n1; i++) {
        leftArr[i] = arr[left + i];
    }
    for (int j = 0; j < n2; j++) {
        rightArr[j] = arr[mid + 1 + j];
    }
    int i = 0, j = 0, k = left;
    while (i < n1 && j < n2) {
        if (leftArr[i] <= rightArr[j]) {
            arr[k] = leftArr[i];
            i++;
        } else {
            arr[k] = rightArr[j];
            j++;
        }
        k++;
    }
    while (i < n1) {
        arr[k] = leftArr[i];
        i++;
        k++;
    }
    while (j < n2) {
        arr[k] = rightArr[j];
        j++;
        k++;
    }
}
void mergeSort(int arr[], int left, int right) {
    if (left < right) {
        // Find the middle point
        int mid = left + (right - left) / 2;

        // Sort first and second halves
        mergeSort(arr, left, mid);
        mergeSort(arr, mid + 1, right);

```



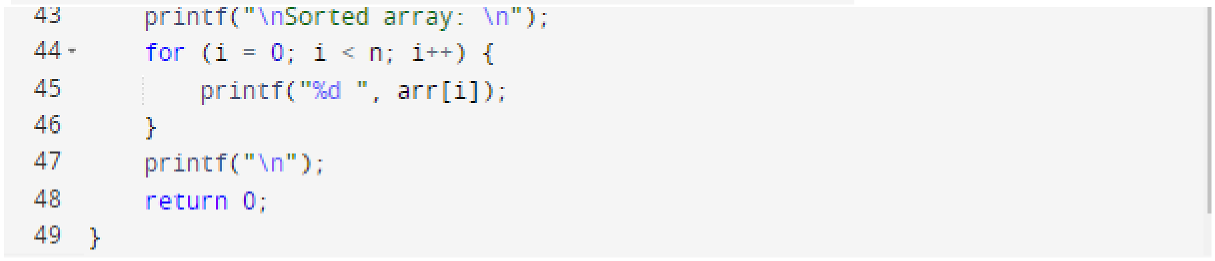
```c
        // Merge the sorted halves
        merge(arr, left, mid, right);
    }
}
int main() {
    int arr[100], n, i;
    printf("Enter the number of elements (maximum 100): ");
    scanf("%d", &n);
    printf("Enter %d elements:\n", n);
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    printf("Unsorted array: \n");
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    mergeSort(arr, 0, n - 1);
    printf("\nSorted array: \n");
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
    return 0;
}
```

BHAVLEEN KAUR
102155010
3EIC-2

7. Write a program to implement Quick sort.



```c
#include <stdio.h>
int partition(int arr[], int low, int high) {
    int pivot = arr[high];
    int i = (low - 1); // index of smaller element

    for (int j = low; j <= high - 1; j++) {
        // If current element is smaller than the pivot
        if (arr[j] <= pivot) {
            i++;
            int temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }
    }
    int temp = arr[i + 1];
    arr[i + 1] = arr[high];
    arr[high] = temp;
    return (i + 1);
}
void quickSort(int arr[], int low, int high) {
    if (low < high) {
        // pi is partitioning index, arr[p] is now at right place
        int pi = partition(arr, low, high);

        // Recursively sort elements before and after partition
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}
int main() {
    int arr[100], n, i;
    printf("Enter the number of elements (maximum 100): ");
    scanf("%d", &n);
    printf("Enter %d elements:\n", n);
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    printf("Unsorted array: \n");
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    quickSort(arr, 0, n - 1);
```



```c
    printf("\nSorted array: \n");
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
    return 0;
}
```