Design & Analysis of
Algorithms
Pranav.N
CH.SC.U4CSE24236
1.Bubble Sort (Code & Output)

```
#include <stdio.h>
int main() {
   printf("Pranav.N\n");
   printf("CH.SC.U4CSE24236\n\n");
   int n;
   printf("Enter number of elements: ");
   scanf("%d", &n);
   int arr[100];
   printf("Enter %d elements: ", n);
   for(int i = 0; i < n; i++) {
      scanf("%d", &arr[i]);
   }
   printf("Original array: ");
   for(int i = 0; i < n; i++) {
      printf("%d ", arr[i]);
   }
   printf("\n");

   // Bubble Sort
   for(int i = 0; i < n-1; i++) {
      for(int j = 0; j < n-i-1; j++) {
         if(arr[j] > arr[j+1]) {
            // swap
            int temp = arr[j];
            arr[j] = arr[j+1];
            arr[j+1] = temp;
         }
      }
   }
   printf("Sorted array: ");
   for(int i = 0; i < n; i++) {
      printf("%d ", arr[i]);
   }
   printf("\n");

   return 0;
}
```

```
Pranav.N
CH.SC.U4CSE24236

Enter number of elements: 5
Enter 5 elements: 99 45 23 78 12
Original array: 99 45 23 78 12
Sorted array: 12 23 45 78 99
```

Interpretation:
Successive comparison of adjacent maximum elements allows us to sort the array layer by layer, yielding a sorted array when the outer iteration concludes.

2.Insertion Sort (Code & Output)

```
#include<stdio.h>
void insertion(int arr[],int n){
int i,j;
for(i=0;i<=n-1;i++){
j=i;
while(j>0 && arr[j-1]>arr[j]){
int temp=arr[j-1];
arr[j-1]=arr[j];
arr[j]=temp;
j--;
}
}
printf("\n");
printf("Sorted array:");
for(int i=0;i<n;i++){
printf("%d",arr[i]);
}
}

int main(){
printf("Name: Pranav.N”);
printf("\n");
printf("Roll No: CH.SC.U4CSE24236");
printf("\n");
int n;
printf("Enter the size of the array: ");
scanf("%d",&n);
int arr[n];
printf("Enter the array elements:");
for(int i=0;i<n;i++){
scanf("%d",&arr[i]);
}
printf("Unsorted array:");
for(int i=0;i<n;i++){
```

printf("%d",arr[i]);
}
insertion(arr,n);
printf("\n");
}

```
$ ./insertion
Name: Pranav.N
Roll No: CH.SC.U4CSE24236

Enter the size of the array: 7
Enter the array elements: 64 34 25 12 22 11 90
Unsorted array: 64 34 25 12 22 11 90
Sorted array: 11 12 22 25 34 64 90
```

Interpretation:
As the loop runs, select elements one by one and arrange them at their correct sorted indices through swapping.

3.Selection Sort (Code & Output)

```
#include<stdio.h>

void Selection(int arr[], int n) {
  int i, j, min;
  for(i = 0; i < n-1; i++) {          // changed to n-1
     min = i;
     for(j = i+1; j < n; j++) {     // start from i+1
        if(arr[j] < arr[min])
           min = j;
     }
     // Swap
     int temp = arr[min];
     arr[min] = arr[i];
     arr[i] = temp;
  }

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

```
    }
    printf("\n");
}

int main() {
    printf("Name: Pranav.N\n");
    printf("Roll No: CH.SC.U4CSE24236\n\n");
    int n;
    printf("Enter the size of the array: ");
    scanf("%d", &n);
    int arr[n];
    printf("Enter the array elements: ");
    for(int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    printf("Unsorted array: ");
    for(int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
    Selection(arr, n);
    return 0;
}
```

```
Name:Pranav.N
Roll No: CH.SC.U4CSE24236
Enter the size of the array: 7
Enter the array elements:64 34 25 12 22 11 90
Unsorted array:64 34 25 12 22 11 90
Sorted array:11 12 22 25 34 64 90
```

Interpretation :
Extract the smallest number to the first position, then the second smallest to the next position, and so on, until achieving a completely sorted array.

4.Bucket Sort (Code & Output)

```
#include <stdio.h>
void Insertion(int arr[], int n){
    if (n <= 0)
        return;
    int i, j;
```

Design & Analysis of
Algorithms
Pranav.N
CH.SC.U4CSE24236

```
   int min = arr[0], max = arr[0];
for (i = 1; i < n; i++) {
       if (arr[i] < min) min = arr[i];
       if (arr[i] > max) max = arr[i];
   }
   int range = max - min + 1;
   int count[range];
for (i = 0; i < range; i++) {
       count[i] = 0;
   }
for (i = 0; i < n; i++) {
       count[arr[i] - min]++;
   }
int index = 0;
   for (i = 0; i < range; i++) {
       while (count[i] > 0) {
           arr[index] = i + min;
           index++;
           count[i]--;
       }
   }
printf("\n");
   printf("Sorted array:");
   for (i = 0; i < n; i++) {
       printf("%d ", arr[i]);
   }
}
int main(){
   printf("Name:Pranav.N");
   printf("\n");
   printf("Roll No: CH.SC.U4CSE24236");
   printf("\n");

   int n;
   printf("Enter the size of the array: ");
   scanf("%d",&n);
   int arr[n];
   printf("Enter the array elements:");
   for (int i = 0; i < n; i++) {
       scanf("%d",&arr[i]);
   }
printf("Unsorted array:");
   for (int i = 0; i < n; i++) {
       printf("%d ",arr[i]);
   }
Insertion(arr,n);
   printf("\n");
}
```

Design & Analysis of
Algorithms
Pranav.N
CH.SC.U4CSE24236

```
Name:Pranav.N
Roll No: CH.SC.U4CSE24236
Enter the size of the array: 7
Enter the array elements:64 34 25 12 22 11 90
Unsorted array:64 34 25 12 22 11 90
Sorted array:11 12 22 25 34 64 90
```

Interpretation :
Place numbers of similar range into separate buckets, then by accessing the buckets systematically, the result is a sorted array.

5.Heap Sort (Code & Output)

```
#include<stdio.h>
void heapify(int arr[], int n, int i){
int largest = i;
int left = 2*i + 1;
int right = 2*i + 2;
if(left < n && arr[left] > arr[largest])
largest = left;
if(right < n && arr[right] > arr[largest])
largest = right;
if(largest != i){
int temp = arr[i];
arr[i] = arr[largest];
arr[largest] = temp;
heapify(arr,n,largest);
}
}
void insertion(int arr[], int n){
for(int i=n/2-1;i>=0;i--)
heapify(arr,n,i);
for(int i=n-1;i>=0;i--){
int temp = arr[0];
arr[0] = arr[i];
```

Design & Analysis of
Algorithms
Pranav.N
CH.SC.U4CSE24236

```
arr[i] = temp;
heapify(arr,i,0);
}
printf("\n");
printf("Sorted array:");
for(i=0;i<n;i++)
printf("%d ",arr[i]);
}

int main(){
printf("Name: Pranav.N");
printf("\n");
printf("Roll No: CH.SC.U4CSE24236");
printf("\n");
int n;
printf("Enter the size of the array: ");
scanf("%d",&n);
int arr[n];
printf("Enter the array elements:");
for(int i=0;i<n;i++){
scanf("%d",&arr[i]);
}
printf("Unsorted array:");
for(int i=0;i<n;i++){
printf("%d ",arr[i]);
}
insertion(arr,n);
printf("\n");
}
```

```
Name: Pranav.N
Roll No: CH.SC.U4CSE24236
Enter the size of the array: 7
Enter the array elements:64 34 25 12 22 11 90
Unsorted array:64 34 25 12 22 11 90
Sorted array:11 12 22 25 34 64 90
```

Interpretation :

Design & Analysis of
Algorithms
Pranav.N
CH.SC.U4CSE24236
The process begins with max-heap construction to elevate the largest element to the top, followed by successive removal and placement of this element at the array's end, resulting in sorted order.