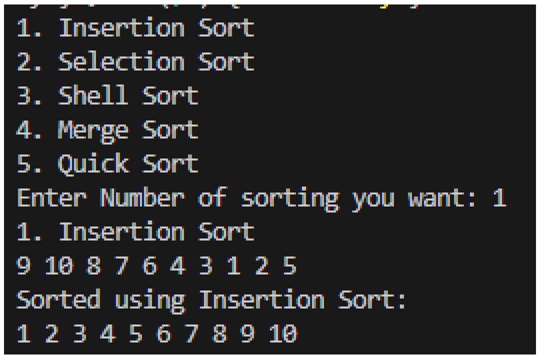
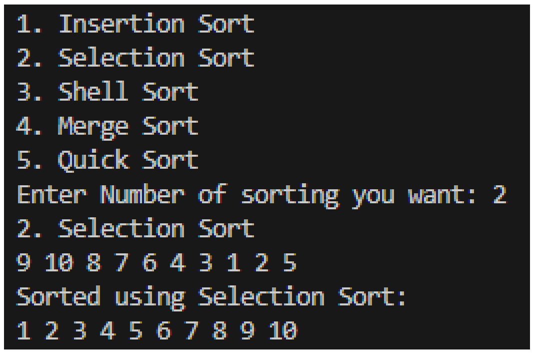
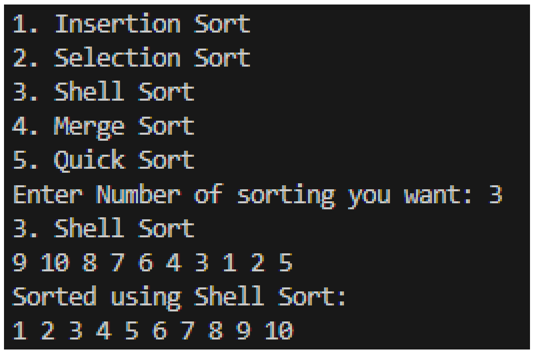
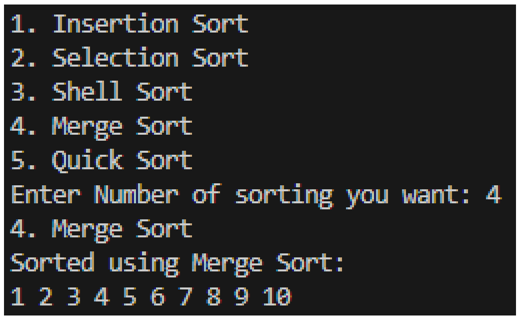
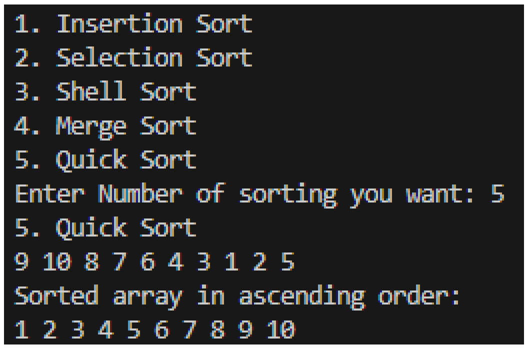
```cpp
//Esteron,Jenel F.
//CPE21S1
#include<iostream>

using namespace std;

int arr[]={9,10,8,7,6,4,3,1,2,5};
int i;
int size = sizeof(arr) / sizeof(arr[0]);

void printArray1(){
    for(int i=0;i<10;i++){
        cout<<arr[i]<<" ";
    }
    cout<<endl;
}
void printArray(int arr[], int size) {
  for (int i = 0; i < size; i++)
    cout << arr[i] << " ";
  cout << endl;
}
void insertion_sort(int arr[],int length){
    printArray1();
    for(int i=1;i<length;i++){
        int key=arr[i];
        int j=i-1;
        while(j>=0 && arr[j]>key){
            arr[j+1]=arr[j];
            j=j-1;
        }
        arr[j+1]=key;
    }
    cout<<"Sorted using Insertion Sort: \n";
    for(int i=0;i<10;i++)
        cout<<arr[i]<<" ";
}
void selection_sort(int arr[],int length){
    for(int i=0;i<10;i++){
        cout<<arr[i]<<" ";
    }
  cout<<"\n";
  for(int i=0;i<10;i++){
    int smallest=arr[i];
    int smallestIndex=i;
    for(int m=i+1;m<10;m++){
      if(arr[m]<smallest){
        smallest=arr[m];
        smallestIndex=m;
      }
    }
    swap(arr[i],arr[smallestIndex]);
  }
  cout<<"Sorted using Selection Sort: \n";
  for(int i=0;i<10;i++){
    cout<<arr[i]<<" ";
  }
}
void shell_sort(int arr[], int length){
  printArray1();
  for (int interval = length / 2; interval > 0; interval /= 2) {
  for (int i = interval; i < length; i += 1) {
   int temp = arr[i];
   int j;
   for (j = i; j >= interval && arr[j - interval] > temp; j -=
interval) {
    arr[j] = arr[j - interval];
   }
   arr[j] = temp;
  }
 }
 cout<<"Sorted using Shell Sort: \n";
  for(int i=0;i<10;i++){
    cout<<arr[i]<<" ";
  }
}
void merge(int arr[], int p, int q, int r) {

 int n1 = q - p + 1;
 int n2 = r - q;
```

```
  int L[n1], M[n2];

  for (int i = 0; i < n1; i++)
    L[i] = arr[p + i];
  for (int j = 0; j < n2; j++)
    M[j] = arr[q + 1 + j];

  int i, j, k;
  i = 0;
  j = 0;
  k = p;
  while (i < n1 && j < n2) {
    if (L[i] <= M[j]) {
      arr[k] = L[i];
      i++;
    } else {
      arr[k] = M[j];
      j++;
    }
    k++;
  }
  while (i < n1) {
    arr[k] = L[i];
    i++;
    k++;
  }

  while (j < n2) {
    arr[k] = M[j];
    j++;
    k++;
  }
}
void merge_sort(int arr[], int l, int r){
    if (l < r) {
    int m = l + (r - l) / 2;

    merge_sort(arr, l, m);
    merge_sort(arr, m + 1, r);
    merge(arr, l, m, r);
}
}
void swap(int *a, int *b) {
  int t = *a;
  *a = *b;
  *b = t;
}
int partition(int array[], int low, int high) {

  int pivot = array[high];

  int i = (low - 1);

  for (int j = low; j < high; j++) {
    if (array[j] <= pivot) {
      i++;

      swap(&array[i], &array[j]);
    }
  }
  swap(&array[i + 1], &array[high]);

  return (i + 1);
}
void quick_sort(int array[], int low, int high){
    if (low < high) {
    int pi = partition(array, low, high);
    quick_sort(array, low, pi - 1);
    quick_sort(array, pi + 1, high);
    int n = sizeof(arr) / sizeof(arr[0]);
    }
}
int main(){
    int Menu;

    cout<<"1. Insertion Sort\n";
    cout<<"2. Selection Sort\n";
    cout<<"3. Shell Sort\n";
    cout<<"4. Merge Sort\n";
```

```cpp
    cout<<"5. Quick Sort\n";
    cout<<"Enter Number of sorting you want: ";
    cin>>Menu;

    switch(Menu){
        case 1:
        cout<<"1. Insertion Sort\n";
            insertion_sort(arr,10);
            break;
        case 2:
            cout<<"2. Selection Sort\n";
            selection_sort(arr,10);
            break;
        case 3:
            cout<<"3. Shell Sort\n";
            shell_sort(arr,10);
            break;
        case 4:
            cout<<"4. Merge Sort\n";
            merge_sort(arr, 0, size - 1);
            cout << "Sorted using Merge Sort: \n";
            printArray(arr, size);
            break;
        case 5:
            cout<<"5. Quick Sort\n";
            printArray(arr, size);
            quick_sort(arr, 0, size - 1);
            cout << "Sorted array in ascending order: \n";
            printArray(arr, size);
            break;
    }
    return 0;
}
```


```
1. Insertion Sort
2. Selection Sort
3. Shell Sort
4. Merge Sort
5. Quick Sort
Enter Number of sorting you want: 1
1. Insertion Sort
9 10 8 7 6 4 3 1 2 5
Sorted using Insertion Sort:
1 2 3 4 5 6 7 8 9 10
```

```
1. Insertion Sort
2. Selection Sort
3. Shell Sort
4. Merge Sort
5. Quick Sort
Enter Number of sorting you want: 2
2. Selection Sort
9 10 8 7 6 4 3 1 2 5
Sorted using Selection Sort:
1 2 3 4 5 6 7 8 9 10
```

```
1. Insertion Sort
2. Selection Sort
3. Shell Sort
4. Merge Sort
5. Quick Sort
Enter Number of sorting you want: 3
3. Shell Sort
9 10 8 7 6 4 3 1 2 5
Sorted using Shell Sort:
1 2 3 4 5 6 7 8 9 10
```

```
1. Insertion Sort
2. Selection Sort
3. Shell Sort
4. Merge Sort
5. Quick Sort
Enter Number of sorting you want: 4
4. Merge Sort
Sorted using Merge Sort:
1 2 3 4 5 6 7 8 9 10
```



```
1. Insertion Sort
2. Selection Sort
3. Shell Sort
4. Merge Sort
5. Quick Sort
Enter Number of sorting you want: 5
5. Quick Sort
9 10 8 7 6 4 3 1 2 5
Sorted array in ascending order:
1 2 3 4 5 6 7 8 9 10
```