Course Name: A Level (2nd Sem) **Subject:** Data Struture using C++
Topic: Sorting in C++ Continued... **Date:** 13-04-2020

Example1 : Sorting an Array using Bubble Sort.

```
#include <iostream>
using namespace std;
void bubbleSort(int a[], int size);
void print(const int a[], int size);
int main()
{
  const int SIZE = 8;
  int a[] = {8, 4, 5, 3, 2, 9, 4, 1};
  print(a, SIZE);
  cout << endl;
  bubbleSort(a, SIZE);
  print(a, SIZE);
  cout << endl;
}
void bubbleSort(int a[], int size)
{
  bool done = false;                    // terminate if no more swap thru a pass
  int pass = 0;                         // pass number, for tracing
  int temp;                             // use for swapping
  while (!done)
{
    cout << "PASS " << ++pass << " ..." << endl;   // for tracing
    done = true;
    for (int i = 0; i < size - 1; ++i)
    {
      if (a[i] > a[i+1])
      {
        print(a, size); // for tracing
        temp = a[i];
        a[i] = a[i+1];
        a[i+1] = temp;
        done = false;                    // swap detected, one more pass
        cout << "=> ";                   // for tracing
        print(a, size);
        cout << endl;
      }
    }
  }
}
void print(const int a[], int size)
{
  cout << "{";
  for (int i = 0; i < size; ++i)
 {
    cout << a[i];
    if (i < size - 1) cout << ",";
  }
  cout << "} ";
```

```
}
```

Example2 : Sorting an Array using Merge Sort (Recursive Top-Down)

```
#include <iostream>
using namespace std;

void mergeSort(int a[], int size);
void mergeSort(int a[], int iLeft, int iRight, int work[]);
void merge(int a[], int iLeftHalfLeft, int iLeftHalfRight,
        int iRightHalfLeft, int iRightHalfRight, int work[]);

void print(const int a[], int iLeft, int iRight);

int main()
{
  const int SIZE_1 = 8;
  int a1[SIZE_1] = {8, 4, 5, 3, 2, 9, 4, 1};

  print(a1, 0, SIZE_1 - 1);
  cout << endl;
  mergeSort(a1, SIZE_1);
  print(a1, 0, SIZE_1 - 1);
  cout << endl << endl;

  const int SIZE_2 = 13;
  int a2[SIZE_2] = {8, 4, 5, 3, 2, 9, 4, 1, 9, 1, 2, 4, 5};

  print(a2, 0, SIZE_2 - 1);
  cout << endl;
  mergeSort(a2, SIZE_2);
  print(a2, 0, SIZE_2 - 1);
  cout << endl;
}

void mergeSort(int a[], int size)
{
  int work[size];  // work space
  mergeSort(a, 0, size - 1, work);
}

void mergeSort(int a[], int iLeft, int iRight, int work[])
{
  if ((iRight - iLeft) >= 1)
{                                                    // more than 1 elements, divide and sort
                                                     // Divide into left and right half
    int iLeftHalfLeft = iLeft;
    int iLeftHalfRight = (iRight + iLeft) / 2;       // truncate
    int iRightHalfLeft = iLeftHalfRight + 1;
    int iRightHalfRight = iRight;

    mergeSort(a, iLeftHalfLeft, iLeftHalfRight, work);
    mergeSort(a, iRightHalfLeft, iRightHalfRight, work);
     merge(a, iLeftHalfLeft, iLeftHalfRight, iRightHalfLeft, iRightHalfRight, work);
  }
}
```

```
void merge(int a[], int iLeftHalfLeft, int iLeftHalfRight,
          int iRightHalfLeft, int iRightHalfRight, int work[])
{
   int size = iRightHalfRight - iLeftHalfLeft + 1;
   int iResult = 0;
   int iLeft = iLeftHalfLeft;
   int iRight = iRightHalfLeft;
   while (iLeft <= iLeftHalfRight && iRight <= iRightHalfRight)
{
      if (a[iLeft] <= a[iRight])
     {
        work[iResult++] = a[iLeft++];
      }
     else
     {
        work[iResult++] = a[iRight++];
      }
   }

   while (iLeft <= iLeftHalfRight) work[iResult++] = a[iLeft++];
   while (iRight <= iRightHalfRight) work[iResult++] = a[iRight++];

                                                                  // for tracing
   print(a, iLeftHalfLeft, iLeftHalfRight);
   print(a, iRightHalfLeft, iRightHalfRight);
   cout << "=> ";
   print(work, 0, size - 1);
   cout << endl;

   for (iResult = 0, iLeft = iLeftHalfLeft; iResult < size; ++iResult, ++iLeft)
  {
      a[iLeft] = work[iResult];
   }
}

// Print the contents of the given array from iLeft to iRight (inclusive)
void print(const int a[], int iLeft, int iRight)
{
   cout << "{";
   for (int i = iLeft; i <= iRight; ++i)
 {
      cout << a[i];
      if (i < iRight) cout << ",";
   }
   cout << "} ";
}
```