Source Code :

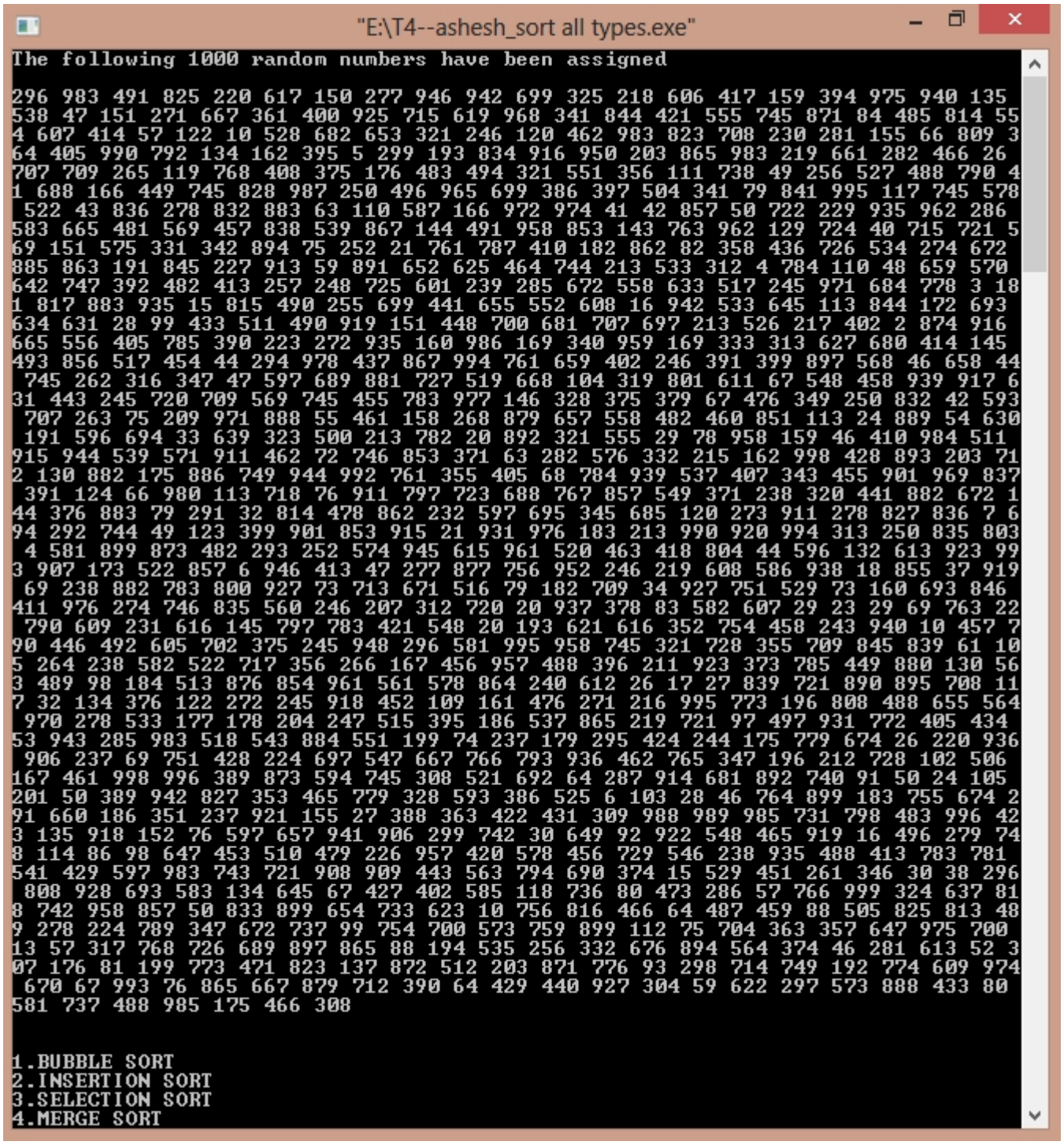
```
#include <stdlib.h>
#include<stdio.h>
int array[1000];
int num=1000,i;
void main()
{
    int inputs=0,ch;
    time_t t;
    printf("The following %d random numbers have been assigned\n\n",num);
    srand(time(&t));
    for(inputs=0;inputs<num;inputs++)
    {
        array[inputs]=rand()%1000;
        printf("%d ",array[inputs]);
    }
    printf("\n\n");
    printf("\n1.BUBBLE SORT\n2.INSERTION SORT\n3.SELECTION SORT\n4.MERGE
SORT\n5.QUICK SORT\n6.EXIT");
    printf("\nPLEASE ENTER YOUR CHOICE\n");
    scanf("%d", & ch);
    switch (ch)
    {
    case 1:
        bubbleSort();
        break;
    case 2:
        insertionSort();
        break;
    case 3:
        selectionSort();
        break;
    case 4:
        {mergeSort(0,num-1);
        printf("\n\nYOU HAVE SELECTED MERGE SORT\nAfter sorting:\n");
        for(i=0; i<num; i++)
        printf("%d ", array[i]);
        printf("\n");
        break;}
    case 5:
        quickSort(0,num-1);
        printf("\n\nYOU HAVE SELECTED QUICK SORT\nAfter sorting:\n");
    for(i=0; i<num; i++)
    printf("%d ", array[i]);
    printf("\n");
        break;
    case 6:
        printf("YOU HAVE SELECTED TO EXIT");
        exit(0);
        break;
    }
}
```

```
bubbleSort()
{
    int count, dim, swap;
    printf("\nYOU HAVE SELECTED BUBBLE SORT\n");
    for (count=0;count<(num-1);count++)
    {
        for (dim=0;dim<num-count-1;dim++)
        {
            if (array[dim] > array[dim+1])
            {
                swap = array[dim];
                array[dim] = array[dim+1];
                array[dim+1] = swap;
            }
        }
    }
    printf("\nSorted List is:\n");
    for (count=0;count<num;count++)
    {
        printf("%d ", array[count]);
    }
}
insertionSort()
{
    int count, dim, temp;
    printf("\nYOU HAVE SELECTED INSERTION SORT\n");
    for (count=1;count<= num-1;count++)
    {
        dim = count;
        while (dim>0 && array[dim]<array[dim-1])
        {
            temp = array[dim];
            array[dim] = array[dim-1];
            array[dim-1] = temp;
            dim--;
        }
    }
    printf("\nSorted List is:\n");
    for (count=0;count<=num-1;count++)
    {
        printf("%d ", array[count]);
    }
}
selectionSort()
{
    int count, dim, pos, swap;
    printf("\nYOU HAVE SELECTED SELECTION SORT\n");
    for (count=0;count<(num-1);count++)
    {
        pos = count;
        for(dim=count+1;dim<num;dim++)
        {
            if ( array[pos] > array[dim] )
```

```
            {
                pos = dim;
            }
        }
        if(pos!=count)//For swapping
        {
            swap = array[count];
            array[count] = array[pos];
            array[pos] = swap;
        }
    }
    printf("\nSorted list is:\n");
    for (count=0;count<num;count++)
    {
        printf("%d ", array[count]);
    }
}
mergeSort(int beg, int end)
{
    int mid;
    if (beg < end)
    {
        mid = (beg+end)/2;
        mergeSort( beg, mid);
        mergeSort( mid+1, end);
        merge_array(beg, mid, end);
    }
}
void merge_array(int beg, int mid, int end)
{
    int i, left_end, num, temp, j, k, b[50];
    for(i=beg; i<=end; i++)
    b[i] = array[i];
    i = beg;
    j = mid+1;
    k = beg;
    while ((i<=mid) && (j<=end))
    {
        if (b[i] <= b[j])
        {
            array[k] = b[i];
            i++;
            k++;
        }
        else
        {
            array[k] = b[j];
            j++;
            k++;
        }
    }
    if (i <= mid)
    {
```

```
        while (i <= mid)
        {
            array[k] = b[i];
            i++;
            k++;
        }
    }
        else
        {
            while (j <= end)
            {
                array[k] = b[j];
                j++;
                k++;
            }
        }
}
quickSort(int beg, int end)
{
    int x;
    if (beg < end)
    {
        x = partition( beg, end);
        quickSort( beg, x-1);
        quickSort( x+1, end);
    }
}
int partition( int beg, int end)
{
    int loc = beg, temp;
    while (1)
    {
        while (array[loc]<=array[end] && loc!=end)
        end--;
        if (loc == end)
        return loc;
        temp = array[loc];
        array[loc] = array[end];
        array[end] = temp;
        loc = end;
        while (array[loc]>=array[beg] && loc!=beg)
        beg++;
        if (loc == beg)
        return loc;
        temp = array[loc];
        array[loc] = array[beg];
        array[beg] = temp;
        loc = beg;
    }
}
```

Sample Input :



Sample Output :