27/8Sept

01

Name: Kshitij Ganvir
Reg no: 20191BCS0005
Roll no: A-04
Sub: OS

- Implement the sort using function. Input data can be 1000 random integers. Write the script in following order.
  - Bubble sort
  - heap "
  - insert "
  - merge "
  - quick sort
  - selection "
  - shell "
  - radix "

(1) Bubble sort: It is a simple sorting algorithm that repeatedly steps through the list to be sorted, compares each pair of adjacent items and swaps them if they are in wrong order

```
# include <stdio.h>
# include <stdlib.h>
void bubblesort (int a[], int size);
void main
{ int a [1000], n, i;
  printf ("Enter size of array");
  scanf ("%d", &n);
if (n > 1000)
  { printf ("error");
    exit (0);
}
```

```
printf(" Enter the array element ");
for (i=0; i<n; i++)
    scanf ("%d", &a[i]);
    bubblesort (a, n);
    printf("the sorted array is ");
    for (i=0; i<n; i++)
    printf ("%d\t", a[i]);
}
void bubblesort (int a[], int size)
{
    int temp, i, j;
    for (i=0; i<size; i++)
    { for (j=0; j<size-1; j++)
    { if (a[j] > a[j+1])
    { temp = a[j];
        a[j] = a[j+1];
        a[j+1] = temp;
    }
}
}
}
```

(2) Heap sort: Heap sort is a comparision based sorting technique based on Binary Heap data structure.

```
# include <iostream>
using namespace std;
void heapify (int arr[], int n, int i)
{
```

```
int largest = i;
int l = 2*i + 1;
int r = 2*i + 2;

if (l < n && arr[l] > arr[largest])
    largest = l;
if (r < n && arr[r] > arr[largest])
    largest = r;
if (largest != i){
    swap (arr[i], arr[largest]);
    heapify (arr, n, largest);
    }
} // main func" to do heap sort
void heapsort (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--){
    swap (arr[0], arr[i]);
    heapify (arr, i, 0);
  }
}
void printarray (int arr[], int n)
 { for (int i=0; i<n; ++i)
    cout<< arr[i] << " ";
    cout<< "\n";
}
int main()
{ int arr[] = { 12, 11, 13, 5, 6, 7 };
  int n = sizeof(arr) / sizeof(arr[0]);
  heapsort (arr, n);
  cout << "sorted array is \n";
  print array (arr, n);
}
```

(3) Insertion Sort:

```
# include <stdio.h>
# include <conio.h>
void insertion (int [], int );
int main ()
{ int arr [30];
  int i, size;
printf (" enter total no. of element : ");
  scanf ("%d" &size);
for (i=0; i<size ; i++)
{ printf (" Enter %d elements: ");
  scanf ("%d", &size);
  for (i=0; i<size ; i++)
  {
    printf (" enter %d element:" i+1);
    scanf ("%d", &arr [i]);
  }
  insertion (arr, size);
  for (i=0; i<size ; i++)
    printf ("%d", arr[i]);
  getch;
  return 0;
}
void insertion (int arr[], int size)
{ int i, j, tmp;
  for (i=0; i<size; i++)
  {
    for (j= i-1 ; j>=0; j--)
  { if ( arr [j] > arr [j+1])
  { tmp = arr [j];
    arr [j] = arr [j+1];
```

```
arr[j+1] = temp;
} else break
}
}
}
```

(4) Merge sort :

```
#include <stdio.h>
#include <stdlib.h>
void merge (int arr[], int l, int m, int r)
{
    int i, j, k;
    int n1 = m - l + 1;
    int n2 = r - m;
int L[n1], R[n2];
for (i = 0; i < n1; i++)
    L[i] = arr[l+i];
for (j = 0; j < n2; j++)
    R[j] = arr[m+1+j];
i = 0;
j = 0;
k = l;
while (i < n1 && j < n2){
    if (L[i] <= R[j]){
        arr[k] = L[i];
        i++;
    } el
    else {
        arr[k] = R[j];
        j++;
    }
    k++;
}
```

```
void mergesort (int arr[], int l, int r)
{ if (l<r){
    int m = l + (r-1)/2;
    mergesort (arr, l, m);
    "          (arr, m+1, r);
    merge (arr, l, m, r);
    }
}
void printArray (int A[], int size)
{   int i;
    for (i=0; i<size; i++)
        printf ("%d", A[i]);
    printf ("\n");
}
int main()
{
  int arr [] = {12, 11, 13, 5, 6, 7};
  int arr-size = size of (arr) / size of (arr[0]);
  printf (" arr , arr-size);
   mergesort (arr, 0, arr-size.-1);
   print f(" Sorted array: ");
   printArray (arr, arr-size);
   return 0;
  }
```

(5) Quick Sort:

```
#  include <studio.h>
   void quicksort (int num[25], first, int last){
    int i, j, pivot, temp;
```

```
if (first < last){
    pivot = first;
    i = first;
    j = last;

    while (i<j){
        while (num[j] > num[pivot])
            j--;
    }
}
    temp = num[pivot];
    num[pivot] = num[j];
    num[j] = temp;
    quicksort (num, j+1, last);
    }
}
int main (){
    int i, count, num[25];
    printf (" no. of elements to enter");
    scanf ("%d", &count);
    printf (" enter %d element", count);
    for (i=0; i<count; i++)
        scanf ("%d", &num[i]);

    quicksort (num, 0, count-1);
    printf (" Order of sorted element:");
    for (i=0; i<count; i++)
        printf ("%d", num[i]);
    return 0;
}
```

(6) Selection Sort:

```
#include <stdio.h>
int main()
{ int a[100], n, i, j, posit^n, swap;
  printf("enter no. of elements");
  scanf("%d", &n);
  printf("enter %d numbers\n", n);
  for (i=0; i<n; i++)
  scanf("%d", &a[i]);
  for (i=0; i<n-1; i++)
  { posit^n = i;
    for (j=i+1; j<n; j++)
  {
      if (a[posit^n] > a[j])
        posi^n = j;
    }
      if (posit^n != i)
      { swap = a[i];
        a[i] = a[posi^n];
        a[posi^n] = swap;
      }
  }
  printf("sorted array: \n");
  for (i=0; i<n; i++)
      printf("%d", a[i]);
  return 0;
}
```

Date
Page 90

27/Sept Ishilj Ganvir 09

(7) Shell Sort:

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

int shellSort (int arr[], int n)
{
    for (int gap = n/2; gap > 0; gap /= 2).
{
    for (int i = gap; i < n; i += 1).
    {
        int temp = arr[i];
        int j;
    for (j = i; j >= gap && arr[j-gap] > temp;
            j -= gap)
    arr[j] = arr[j] - gap];
    arr[j] = temp;
    }
}
    return 0;
}
int main ()
{ int arr[] = {12,34, 54, 2, 3}; i;
    int n = sizeof(arr) / sizeof(arr[0]);
    cout << " array before sorting: n";
    Shell sort (arr, n);
    cout << " array after sorting";
    printArray (arr, n);
    return 0;
}
```

8} Radix Sort

```
# include <iostream>
using name space std;

int getmax( int arr[], int n)
 {
    int mx = arr[0];
    for (int i = 1; i <n; i++)
        if (arr[i] > mx)
            mx = arr[i];
        return mx;
}
  void countsort (int arr[], int n, int exp)
{
     int output [n];
     int i, count [10] = {0};

   for(i=0; i<n; i++)
     count [(arr[i] /exp) % 10] ++;

   for (i = 1; i< 10; i ++)
       count [i] + = count [i-1];

   for (i = n-1; i >=0; i--) {
       output [count [(arr[i]/exp % 10] - 1] = arr[i];
           count [arr[i] / exp ) % 10] --;
 }
       for (i=0); i <n; i++)
           arr[i] = output [i];
}
```

voi

```
void radixsort (int arr[], int n)
{
    int m = getmax (arr, n);
    for (int exp = 1; m/ exp > 0; exp* = 10)
        countsort (arr, n, exp);
}
    void print (int arr[], int n)
{
    for (int i = 0; i < n; i++)
        cout << arr [i] << " ";
}
    int main ()
{
    int arr[] = { 170, 45, 75, 90, 802, 24, 2, 66 };
    int n = sizeof (arr) / sizeof (arr [0]);
        radixsort (arr, n);
        print (arr, n);
        return 0;
}
```

27/Sept/21

Kshitij Ganvir