V.Venkat raja

AP19110010420

CSE F

LAB PROGRAMS

1)/* Insertion sort ascending order */

#include <stdio.h>

int main()

{

 int n, array[1000], c, d, t, flag = 0;

 printf("Enter number of elements\n");

 scanf("%d", &n);

 printf("Enter %d integers\n", n);

 for (c = 0; c < n; c++)

  scanf("%d", &array[c]);

 for (c = 1 ; c <= n - 1; c++) {

  t = array[c];

  for (d = c - 1 ; d >= 0; d--) {

   if (array[d] > t) {

    array[d+1] = array[d];

    flag = 1;

   }

```c
        else

            break;

    }

    if (flag)

        array[d+1] = t;

}


    printf("Sorted list in ascending order:\n");


    for (c = 0; c <= n - 1; c++) {

        printf("%d\n", array[c]);

    }


    return 0;

}
```

2) // Selection sort in C

```c
#include <stdio.h>


// function to swap the the position of two elements
void swap(int *a, int *b) {

    int temp = *a;

    *a = *b;

    *b = temp;

}


void selectionSort(int array[], int size) {

    for (int step = 0; step < size - 1; step++) {

        int min_idx = step;
```

```c
    for (int i = step + 1; i < size; i++) {

      // To sort in descending order, change > to < in this line.
      // Select the minimum element in each loop.
      if (array[i] < array[min_idx])
        min_idx = i;
    }

    // put min at the correct position
    swap(&array[min_idx], &array[step]);
  }
}

// function to print an array
void printArray(int array[], int size) {
  for (int i = 0; i < size; ++i) {
    printf("%d  ", array[i]);
  }
  printf("\n");
}

// driver code
int main() {
  int data[] = {20, 12, 10, 15, 2};
  int size = sizeof(data) / sizeof(data[0]);
  selectionSort(data, size);
  printf("Sorted array in Acsending Order:\n");
  printArray(data, size);
}
```

3) /* Implementing Bubble sort in a C Program

* Written by: Chaitanya.
*/

```c
#include<stdio.h>

int main(){

  int count, temp, i, j, number[30];

  printf("How many numbers are u going to enter?: ");
  scanf("%d",&count);

  printf("Enter %d numbers: ",count);

  for(i=0;i<count;i++)
  scanf("%d",&number[i]);

  /* This is the main logic of bubble sort algorithm
   */
  for(i=count-2;i>=0;i--){
    for(j=0;j<=i;j++){
      if(number[j]>number[j+1]){
        temp=number[j];
        number[j]=number[j+1];
        number[j+1]=temp;
      }
    }
  }
```

```c
    printf("Sorted elements: ");

  for(i=0;i<count;i++)

    printf(" %d",number[i]);


  return 0;

}
```

4) /* C program for Merge Sort */

```c
#include<stdlib.h>

#include<stdio.h>


// Merges two subarrays of arr[].

// First subarray is arr[l..m]

// Second subarray is arr[m+1..r]

void merge(int arr[], int l, int m, int r)

{

        int i, j, k;

        int n1 = m - l + 1;

        int n2 = r - m;


        /* create temp arrays */

        int L[n1], R[n2];


        /* Copy data to temp arrays L[] and R[] */

        for (i = 0; i < n1; i++)

                L[i] = arr[l + i];

        for (j = 0; j < n2; j++)

                R[j] = arr[m + 1+ j];
```

```
/* Merge the temp arrays back into arr[l..r]*/

i = 0; // Initial index of first subarray

j = 0; // Initial index of second subarray

k = l; // Initial index of merged subarray

while (i < n1 && j < n2)

{

        if (L[i] <= R[j])

        {

                arr[k] = L[i];

                i++;

        }

        else

        {

                arr[k] = R[j];

                j++;

        }

        k++;

}


/* Copy the remaining elements of L[], if there
are any */

while (i < n1)

{

        arr[k] = L[i];

        i++;

        k++;

}
```

```c
        /* Copy the remaining elements of R[], if there

        are any */

        while (j < n2)

        {

                arr[k] = R[j];

                j++;

                k++;

        }

}


/* l is for left index and r is right index of the

sub-array of arr to be sorted */

void mergeSort(int arr[], int l, int r)

{

        if (l < r)

        {

                // Same as (l+r)/2, but avoids overflow for

                // large l and h

                int m = l+(r-l)/2;

                // Sort first and second halves

                mergeSort(arr, l, m);

                mergeSort(arr, m+1, r);

                merge(arr, l, m, r);

        }

}


/* UTILITY FUNCTIONS */
```

```c
/* Function to print an array */
void printArray(int A[], int size)
{
        int i;
        for (i=0; i < size; i++)
                printf("%d ", A[i]);
        printf("\n");
}


/* Driver program to test above functions */
int main()
{
        int arr[] = {12, 11, 13, 5, 6, 7};
        int arr_size = sizeof(arr)/sizeof(arr[0]);

        printf("Given array is \n");
        printArray(arr, arr_size);

        mergeSort(arr, 0, arr_size - 1);

        printf("\nSorted array is \n");
        printArray(arr, arr_size);
        return 0;
}
```