```
/*Estructuras de datos - LIS 2032-1
Nombre Completo: Diana Laura Reyes Youshimatz
ID: 173391
Descripción breve del programa:
Programa con implementación de 5 metodos de oordenamiento
y la opcion de imprimir y salir.
Metodos desarrollados:
selecion sort
bubble sort
merge sort
quick sort
radix sort
*/
#include<stdio.h>
#include<stdlib.h>

void display(int arr[],int n){ int i;
    for(i=0;i<n;i++){
        printf(" %d ", arr[i]);
  }
}
void bubble_sort(int arr[],int n){
  int i,j,temp;
  for(i=0;i<n;i++){
      for(j=0;j<n-i-1;j++){
          if(arr[j]>arr[j+1]){
```

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

}
void selection_sort(int arr[],int n){
    int i,j,temp;
    for(i=0;i<n-1;i++){
        for(j=i+1;j<n;j++){
            if(arr[i]>arr[j]){
             temp=arr[i];
             arr[i]=arr[j];
             arr[j]=temp;
            }
        }
    }
}
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++;
        }
        else{
            arr[k]=R[j];
            j++;
        }
        k++;
    }
    while (i<n1) {
        arr[k]=L[i];
        i++;
        k++;
    }
    while (j<n2) {
        arr[k]=R[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 arr[], int l, int h){
            int pivot=arr[h];
            int i=(l-1);
            int j;
            for (j=l; j<h; j++){
                if (arr[j]<=pivot){
                  i++;
                  swap(&arr[i], &arr[j]);
                }
  }
  swap(&arr[i+1], &arr[h]);
  return (i+1);
}
int get_max(int array[], int n){
  int max=array[0]; int i;
  for (i=1; i<n; i++)
    if (array[i]>max)
```

```
        max=array[i];
  return max;
}

void contar(int arr[], int size, int p){
  int res[size+1];
  int max=(arr[0]/p) % 10;
int i;
  for (i=1; i<size; i++){
    if (((arr[i]/p)%10)>max)
      max=arr[i];
  }
  int c[max+1];

  for (i=0;i<max; ++i)
    c[i]=0;

  for (i=0; i<size; i++)
    c[(arr[i]/p) % 10]++;

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

  for (i=size-1; i>=0; i--){
    res[c[(arr[i]/p) % 10] - 1]=arr[i];
    c[(arr[i]/p) % 10]--;
  }
```

```
  for (i=0; i<size; i++)
    arr[i]=res[i];
}

void radix_sort(int arr[], int size){
  int max=get_max(arr, size); int p;
  for (p=1; max/p>0; p*=10)
    contar(arr, size, p);
}

void quick_sort(int arr[], int l, int h){
  if (l<h){
    int pi=partition(arr, l, h);
    quick_sort(arr, l, pi 1);
    quick_sort(arr, pi + 1, h);
  }
}




int main(){
   int n,choice,i;
   char ch[20];
   printf("Numero de elementos a ordenar: ");
   scanf("%d",&n);
   int arr[n];
   for(i=0;i<n;i++)
```

```
  {
      printf("Ingresa el elemento %d: ",i+1);
      scanf("%d",&arr[i]);
  }
  printf("Opciones:  \n");
  printf("\n1. Bubble Sort\n2. Selection Sort\n3. Merge Sort\n4. Quick sort \n5. Radix sort \n6. Imprimir arreglo\n7. Salir\n");
while(1){



   printf("\nIngresa el numero de la opcion: ");
   scanf("%d",&choice);


   switch(choice)
   {
   case 1:
         printf("Arreglo: ");
         display(arr,n);
      bubble_sort(arr,n);
      printf("Ordenado bubble sort: ");
      display(arr,n);
display(arr,n);
      break;
   case 2:
         printf("Arreglo: ");
         display(arr,n);
      selection_sort(arr,n);
      printf("Ordenado selection sort: ");
      display(arr,n);
```

```
display(arr,n);
        break;
    case 3:
                    merge_sort(arr, 0,n-1);
                    printf("Ordenado merge sort: ");
                    display(arr,n);
        break;
    case 4:
        printf("Arreglo: ");
        display(arr,n);
                    quick_sort(arr, 0,n-1);
                    printf("Ordenado quick sort: ");
                    display(arr,n);
        break;
    case 5:
        printf("Arreglo: ");
        display(arr,n);
                    radix_sort(arr,n);
                    printf("Ordenado radix sort: ");
                    display(arr,n);
        break;
    case 6:
        display(arr,n);
        break;

    case 7:
        return 0;
    default:
```

```
        printf("\n Opcion invalida\n");
    }
}
return 0;
}
```