Lab Cycle #7

1. Creating a library of sorting and searching functions and writing a menu-driven program that uses them

Instructions:

1. Create a C file, senso.h which contains the definitions of the functions:
   a) linearsearch()
   b) binarysearch()
   c) bubblesort()
   d) insertionsort()
   e) selectionsort() and
   f) mergesort()

2. Save the file.

3. Create another C file, sensomenu.c which displays a menu of searching and sorting techniques and performs a particular searching or sorting method according to the user's choice. You should write #include "senso.h" at the beginning.

   a) The Menu should appear as follows:

      1. Searching
      2. Sorting

      What would you like to do?

   b) If the input is 1, a sub menu should appear as,

      1. Linear Search
      2. Binary Search

      What would you like to do?

      ➔ If the input is 1 then

      i) Read number of elements

ii) Read the elements

iii) Read the key

iv) call linearsearch()

➔ If the input is 2 then

i) Read number of elements

ii) Read the elements

iii) Read the key

iv) call binarysearch()

c) If the input is 2, a sub menu should appear as,

1. Bubble Sort
2. Insertion Sort
3. Selection Sort
4. Merge Sort

What would you like to do?

➔ If the input is 1, then

i) Read number of elements
ii) Read the elements
iii) Call bubblesort()
iv) Display the output

➔ If the input is 2, then

i) Read number of elements
ii) Read the elements
iii) Call insertionsort()
iv) Display the output

➔ If the input is 3, then

i) Read number of elements
ii) Read the elements
iii) Call selectionsort()
iv) Display the output

➔ If the input is 4, then

i) Read number of elements
ii) Read the elements
iii) Call bubblesort()
iv) Display the output

4. Procedures

```
Procedure linearsearch(a,key,n)
    1. for i=1 to n do
                if a[i]=key then
                        return(i)
    2.  return(-1)
```

```
Procedure bsearch(a,key,n)
    1. mid=n/2
    2. if a[mid]=key then
                return(mid)
       else if key<a[mid] then
                for i=1 to mid-1 do
                        if a[i]=key then
                                return(i)
                return(-1)
       else if key>a[mid] then
                for i=mid+1 to n do
                        if a[i]=key then
                                return(i)
                return(-1)
```

```
Procedure bubblesort(a, n)
   1. for i=1 to n-1 do
        for j=1 to n-1 do
          if(a[j]>a[j+1])
            swap(a[j],a[j+1])
```

```
Procedure selectionsort(a,n)
  1. for i=1 to n-1 do
                j=i
                for k= i+1 to  n do
                        if(a[j]>=a[k])
                                j=k
                swap(a[i],a[j])
```

```
Procedure insertionsort(a,n)
  1. for i=2 to n do
                key=a[i]
                j=i
                while((j>1) and(key<a[j]) do
                        a[j]=a[j-1]
                        j=j-1
                a[j]=key
```

```
Procedure mergesort(a,i,j)
    1. if(i<j) then
          mid=(i+j)/2
          call mergesort(a,i,mid)
          call mergesort(a,mid+1,j)
          call merge(a,i,mid,j)
```

```
Procedure merge (a,low,mid,high)
    1.  h=low;
     i=low;
     j=mid+1;

    2. while(h<=mid && j<=high)
    {
      if(a[h]<=a[j])
           b[i]=a[h++];
      else
           b[i]=a[j++];
```

```
    i++;
  }
  if( h > mid)
              for(k=j;k<=high;k++)
                        b[i++]=a[k];
  else
              for(k=h;k<=mid;k++)
                        b[i++]=a[k];

  for(k=low;k<=high;k++)
      a[k]=b[k];
}
```