DATA STRUCTURES AND ALGORITHM

ASSIGNMENT 3

NAME: A HARI KRISHNA SWARUN
REG NO 21BPS1252

1. Merge sort

Algorithm:

Void merge(int arr[],int low,int high)
{
Int mid; If(low<high){
Mid=low+high/2;
Merge(arr,low,mid);
Merge(arr,mid+1,high);
Merge(arr,low,mid,high);
}
}

Program Code:

```
#include <stdio.h>

void merge(int arr[], int p, int q, int r) { int

  n1 = q - p + 1;

  int n2 = r - q; int i,j;

  int L[n1], M[n2];


  for (i = 0; i < n1; i++) L[i] = arr[p

    + i];

  for (j = 0; j < n2; j++) M[j] = arr[q

  + 1 + j];
```

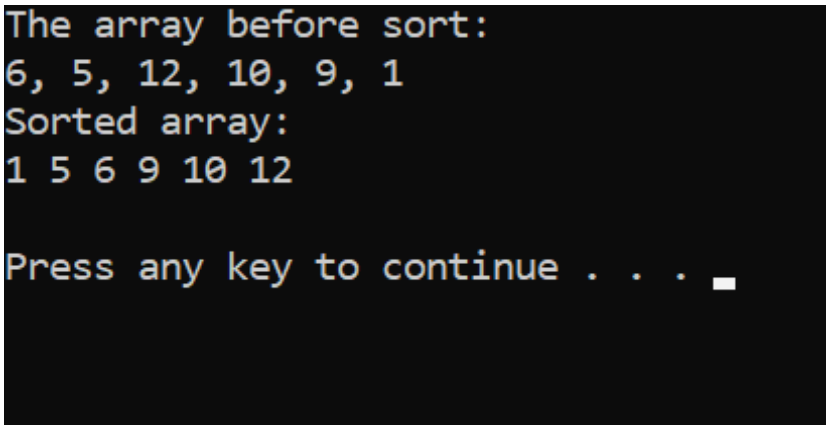
```
  int k;
  i = 0;
  j = 0;
  k = p;
  while (i < n1 && j < n2) {
  if (L[i] <= M[j]) {
     arr[k] = L[i]; i++;
   } else {
     arr[k] = M[j]; j++;
   } k++;
  }
  while (i < n1) { arr[k] =
   L[i]; i++;
   k++;
  }

  while (j < n2) { arr[k] =
   M[j]; j++;
   k++;
  }
}
void mergeSort(int arr[], int l, int r) { if (l
  < r) {
    int m = l + (r - l) / 2;
    mergeSort(arr, l, m);
    mergeSort(arr, m + 1, r);
    merge(arr, l, m, r);
```

```
    }

}
void printArray(int arr[], int size) {
        int i;
  for (i = 0; i < size; i++)
  printf("%d ", arr[i]);
  printf("\n");
}
int main() {
  int arr[] = {6, 5, 12, 10, 9, 1};
  printf("The array before sort:\n");
  printf("6, 5, 12, 10, 9, 1\n");
  int size = sizeof(arr) / sizeof(arr[0]);

  mergeSort(arr, 0, size - 1);

  printf("Sorted array: \n");
  printArray(arr, size);
}
```

```c
#include <stdio.h>
void merge(int arr[], int p, int q, int r)
{
int n1 = q - p + 1;
int n2 = r - q; int i,j;
int L[n1], M[n2];


for (i = 0; i < n1; i++) L[i] = arr[p + i];
for (j = 0; j < n2; j++) M[j] = arr[q + 1 + j];
int k;
i = 0;
j = 0;
k = p;
while (i < n1 && j < n2) {
if (L[i] <= M[j]) {
arr[k] = L[i]; i++;
} else {
arr[k] = M[j]; j++;
} k++;
}
while (i < n1) { arr[k] = L[i]; i++;
k++;
}


while (j < n2) { arr[k] = M[j]; j++;
k++;
}
}
```

```c
void mergeSort(int arr[], int l, int r) {
   if (l < r)
    {
      int m = l + (r - l) / 2; mergeSort(arr, l, m); mergeSort(arr, m + 1, r); merge(arr, l, m, r);
    }
  }
void printArray(int arr[], int size) {
int i;
for (i = 0; i < size; i++) printf("%d ", arr[i]); printf("\n");
}
int main() {
int arr[] = {6, 5, 12, 10, 9, 1};
printf("The array before sort:\n"); printf("6, 5, 12, 10, 9, 1\n");
int size = sizeof(arr) / sizeof(arr[0]);

mergeSort(arr, 0, size - 1);

printf("Sorted array: \n"); printArray(arr, size);
}
```

OUTPUT:



2. Quick sort

Algorithm:

```
Void quick(array, leftmostIndex, rightmostIndex) if
 (leftmostIndex < rightmostIndex)
   pivotIndex <- partition(array,leftmostIndex, rightmostIndex)
   quickSort(array, leftmostIndex, pivotIndex - 1)
   quickSort(array, pivotIndex, rightmostIndex)

partition(array, leftmostIndex, rightmostIndex) set
 rightmostIndex as pivotIndex
 storeIndex <- leftmostIndex - 1
 for i <- leftmostIndex + 1 to rightmostIndex if
 element[i] < pivotElement
   swap element[i] and element[storeIndex]
   storeIndex++
 swap pivotElement and element[storeIndex+1]
return storeIndex + 1
```

Program Code:

```
#include <stdio.h>

void swap(int *a, int *b) { int t = *a;
```

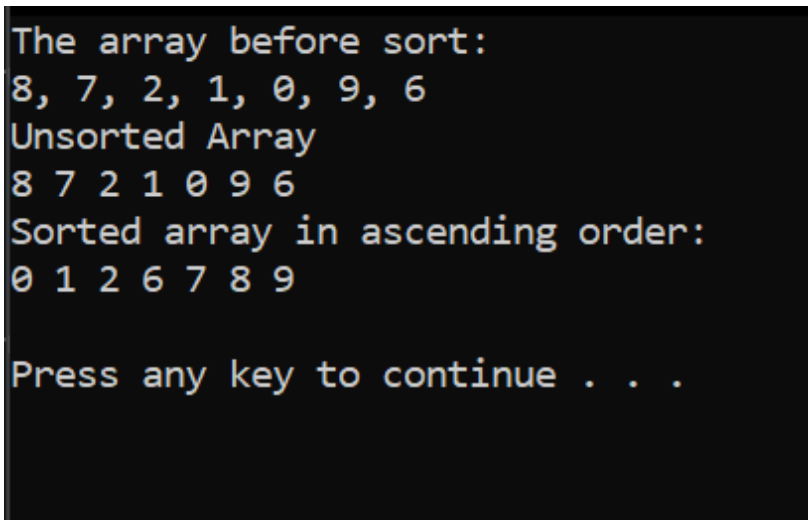
```
  *a = *b;
  *b = t;
}
int partition(int array[], int low, int high) { int pivot
  = array[high];
  int i,j;
  i = (low - 1);
  for (j = low; j < high; j++) { if (array[j]
    <= pivot) { i++;
  swap(&array[i], &array[j]);
    }
  }
  swap(&array[i + 1], &array[high]); return (i
  + 1);
}

void quickSort(int array[], int low, int high) { if (low <
  high) {
    int pi = partition(array, low, high);
    quickSort(array, low, pi - 1);
    quickSort(array, pi + 1, high);
    }

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

```
int main() {
  int data[] = {8, 7, 2, 1, 0, 9, 6};
  printf("The array before sort:\n");
  printf("8, 7, 2, 1, 0, 9, 6\n");
  int n = sizeof(data) / sizeof(data[0]);
   printf("Unsorted Array\n");
  printArray(data, n);
  quickSort(data, 0, n - 1);
  printf("Sorted array in ascending order: \n");
  printArray(data, n);
}
```

```c
#include <stdio.h>
void swap(int *a, int *b) { int t = *a;
*a = *b;
*b = t;
}
int partition(int array[], int low, int high) { int pivot = array[high];
int i,j;
i = (low - 1);
for (j = low; j < high; j++) { if (array[j] <= pivot) { i++;
    swap(&array[i], &array[j]);

}

}
swap(&array[i + 1], &array[high]); return (i + 1);
}


void quickSort(int array[], int low, int high) { if (low < high) {
int pi = partition(array, low, high); quickSort(array, low, pi - 1); quickSort(array, pi + 1, high);
}
}
void printArray(int array[], int size) { int i;
for (i = 0; i < size; ++i) { printf("%d ", array[i]);
}
printf("\n");
}
int main() {
int data[] = {8, 7, 2, 1, 0, 9, 6};
printf("The array before sort:\n");
printf("8, 7, 2, 1, 0, 9, 6\n");
```

```c
int main() {
int data[] = {8, 7, 2, 1, 0, 9, 6};
printf("The array before sort:\n");
printf("8, 7, 2, 1, 0, 9, 6\n");
int n = sizeof(data) / sizeof(data[0]);
 printf("Unsorted Array\n");
printArray(data, n);
quickSort(data, 0, n - 1);
printf("Sorted array in ascending order: \n"); printArray(data, n);
}
```

OUTPUT:


The array before sort:
8, 7, 2, 1, 0, 9, 6
Unsorted Array
8 7 2 1 0 9 6
Sorted array in ascending order:
0 1 2 6 7 8 9
Press any key to continue . . .