```
/*
Aim:                To study and implement divide and conquer sorting methods-
                    Merge sort,Quick sort and Heap sort
Date:               August 7,2019
Author:             Mehul Y Khandhadiya
Problem Statement:  Using the utility functions created in Experiment-2, write a menu driven
                    program to order a list in ascending order using following "divide-and
                    conquer" approaches – the Quick Sort, the Heap Sort, and the Merge Sort.
                    You should compare the running time for ordering lists different input
                    sizes in respect of traditional comparison sorts and divide-and-conquer
                    sorts.
*/
//header files
#include<stdio.h>
#include<time.h>
#include<sortutil.c>
#define MX_SIZE 250000
//function prototypes
void qsort(int*,int,int);
void merge_sort(int*,int,int);
void heap_sort(int*,int);
int ascending_order(int*,int);
int descending_order(int*,int);
int asc_nearsort(int*,int);
int random_order(int*,int);
void copy(int*,int*,int);
void display(int*,int);
//the driver function
int main(){
       int a[MX_SIZE],b[MX_SIZE],num,choice,choice1,n1,n,n2,p=0,r,heap_size;
```

```
double elapsed_t;
time_t t1,t2;
do
{
       printf("Want to create a list?\n Enter 1 for yes:");
       scanf("%d",&num);
       if(num==1)
       {
              do
              {
              printf("\nEnter no. of elements in multiple of 10:");
              scanf("%d",&n);
              }while(n%10!=0);
              heap_size=n-1;
              r=n-1;
              do
              {
              printf("\nWhich type of list you want?\n");
              printf(" 1.ascending 2.descending 3.ascending near sorted 4.random ");
              printf("\nEnter choice:");
              scanf("%d",&choice);
              switch(choice)
              {
              case 1: ascending_order(a,n);
                      printf("\nDo you want to copy the generated array? Enter 1 for
                      yes or any other no. to exit:");
                       scanf("%d",&n2);
                       if(n2==1){
                      copy(a,b,n);
                       }
```

```
        break;

  case 2: descending_order(a,n);

          printf("\nDo you want to copy the generated array? Enter 1 for
          yes or any other no. to exit:");

          scanf("%d",&n2);

          if(n2==1){

          copy(a,b,n);

          }

         break;

  case 3: asc_nearsort(a,n);

          printf("\nDo you want to copy the generated array? Enter 1 for
          yes or any other no. to exit:");

          scanf("%d",&n2);

          if(n2==1){

          copy(a,b,n);

          }

         break;

  case 4: random_order(a,n);

          printf("\nDo you want to copy the generated array? Enter 1 for
          yes or any other no. to exit:");

          scanf("%d",&n2);

          if(n2==1){

          copy(a,b,n);

          }

         break;

default: printf("Invalid input");

         break;

}


do{

        printf("\nWhich type of sorting procedure do you want?\n");
```

```
printf("1.quick sort 2.merge sort 3.heap sort 4.exit \n");
printf("\nEnter choice:");
scanf("%d",&choice1);
switch(choice1)
{
case 1: t1=clock();
        qsort(a,p,r);
        t2=clock();
        elapsed_t=((double)(t2-t1))/CLOCKS_PER_SEC;
        printf("\n\t Time required=%g seconds",elapsed_t);
        printf("\n Do you want to display sorted array?\n
        enter 1 for yes and any other no.to exit");
        scanf("%d",&n1);
        if(n1==1){
        display(a,n);
        }
        break;
case 2: t1=clock();
        merge_sort(a,p,r);
        t2=clock();
        elapsed_t=((double)(t2-t1))/CLOCKS_PER_SEC;
        printf("\n\t Time required=%g seconds",elapsed_t);
        printf("\n Do you want to display sorted array?\n
        enter 1 for yes and any other no.to exit");
        scanf("%d",&n1);
        if(n1==1){
        display(a,n);
        }
        break;
case 3: t1=clock();
```

```
                                heap_sort(a,heap_size);
                                t2=clock();
                                elapsed_t=((double)(t2-t1))/CLOCKS_PER_SEC;
                                printf("\n\t Time required=%g seconds",elapsed_t);
                                printf("\n Do you want to display sorted array?\n
                                enter 1 for yes and any other no.to exit");
                                scanf("%d",&n1);
                                if(n1==1){
                                display(a,n);
                                }
                               break;
                        case 4: printf("terminating condition entered\n");
                                exit(0);
                                 }
                          }while(choice1!=0);
             }while(choice!=0);
       }
       else
       printf("\nTerminating\n ");
       }while(num!=0);
    return 0;
}
void qsort(int a[],int p,int r)
{
int q;
if(p<r)
{
       q=partition(a,p,r);
       qsort(a,p,q-1);
       qsort(a,q+1,r);
```

```
}
}
int partition(int a[],int p,int r)
{
int pivot,temp,i,j,temp1;
pivot=a[r];
i=p-1;
for(j=p;j<r;j++){
if(a[j]<=pivot)
{
       i=i+1;
       temp=a[i];
       a[i]=a[j];
       a[j]=temp;
}
}
temp1=a[i+1];
a[i+1]=a[r];
a[r]=temp1;
return (i+1);
}
void merge_sort(int a[],int p,int r)
{
int q;
if(p<r)
{
       q=(p+r)/2;
       merge_sort(a,p,q);
       merge_sort(a,q+1,r);
       merge(a,p,q,r);
```

```
}
}
void merge(int a[],int p,int q,int r){
int n1,n2,i,j,l[MX_SIZE],m[MX_SIZE],k;
n1=(q-p+1);
n2=(r-q);
for(i=0;i<n1;i++){
       l[i]=a[p+i];
}
for(j=0;j<n2;j++){
       m[j]=a[q+j+1];
}
l[n1]=423256;
m[n2]=423256;
i=0,j=0;
for(k=p;k<=r;k++){
if(l[i]<=m[j]){
       a[k]=l[i];
       i=i+1;
}
else
{
       a[k]=m[j];
       j=j+1;
}
}
}
int parent(int i)
{
       return(i/2);
```

```
}
int left(int i)
{
	return((2*i)+1);
}
int right(int i)
{
	return((2*i)+2);
}
void build_heap(int a[],int heap_size)
{
int i;
for(i=heap_size/2;i>=0;i--)
{
	heapify(a,heap_size,i);
}
}
void heapify(int a[],int heap_size,int i)
{
int l,r,largest,temp;
l=left(i);
r=right(i);
if((l<=heap_size) && (a[l]>a[i]))
{
	largest=l;
}
else
{
	largest=i;
}
```

```
if((r<=heap_size) && (a[r]>a[largest]))
{
       largest=r;
}
if(largest!=i)
{
       temp=a[i];
        a[i]=a[largest];
        a[largest]=temp;
        heapify(a,heap_size,largest)}
}
void heap_sort(int a[],int heap_size)
{
int i,temp;
build_heap(a,heap_size);
for(i=heap_size;i>0;i--)
{
       temp=a[i];
        a[i]=a[0];
        a[0]=temp;
        heap_size--;
        heapify(a,heap_size,0);
}
}
```

//Execution Trail

```
Enter no. of elements in multiple of 10:10
Which type of list you want?
1.ascending 2.descending 3.ascending near sorted 4.random
Enter choice:1
Do you want to copy the generated array?Enter 1 for yes or any other no. to exit:1
```

```
Enter 1 if you want to display copied array otherwise 0:1
14109  14114  14119  14124  14129  14134  14139  14144  14149  14154
Which type of sorting procedure do you want?
1.quick sort 2.merge sort 3.heap sort 4.exit
Enter choice:1
Time required=2e-06 seconds
Do you want to display sorted array?
enter 1 for yes and any other no.to exit1
14109  14114  14119  14124  14129  14134  14139  14144  14149  14154
Which type of sorting procedure do you want?
1.quick sort 2.merge sort 3.heap sort 4.exit
Enter choice:2
Do you want to display sorted array?
Which type of sorting procedure do you want?
1.quick sort 2.merge sort 3.heap sort 4.exit
Enter choice:3
Time required=5e-06 seconds
Which type of sorting procedure do you want?
1.quick sort 2.merge sort 3.heap sort 4.exit
Enter choice:4
terminating condition entered
Enter no. of elements in multiple of 10:10
Which type of list you want?
1.ascending 2.descending 3.ascending near sorted 4.random
Enter choice:2
Do you want to copy the generated array?Enter 1 for yes or any other no. to exit:1
Enter 1 if you want to display copied array otherwise 0:1
14109   14104   14099   14094   14089   14084   14079   14074   14069   14064
Which type of sorting procedure do you want?
1.quick sort 2.merge sort 3.heap sort 4.exit
```

```
Enter choice:1
Time required=2e-06 seconds
Do you want to display sorted array?
enter 1 for yes and any other no.to exit1
14064   14069   14074   14079   14084   14089   14094   14099   14104   14109
Which type of sorting procedure do you want?
1.quick sort 2.merge sort 3.heap sort 4.exit
Enter choice:2
Time required=2.6e-05 seconds
Which type of sorting procedure do you want?
1.quick sort 2.merge sort 3.heap sort 4.exit
Enter choice:3
Time required=4e-06 seconds
Which type of sorting procedure do you want?
1.quick sort 2.merge sort 3.heap sort 4.exit
Enter choice:4
terminating condition entered
Enter no. of elements in multiple of 10:10
Which type of list you want?
1.ascending 2.descending 3.ascending near sorted 4.random
Enter choice:3
Do you want to copy the generated array? Enter 1 for yes or any other no. to exit:1
Enter 1 if you want to display copied array otherwise 0:1
14109   14114   14119   14130   14129   14134   14151   14144   14149   14172
Which type of sorting procedure do you want?
1.quick sort 2.merge sort 3.heap sort 4.exit
Enter choice:1
Time required=3e-06 seconds
Do you want to display sorted array?
enter 1 for yes and any other no.to exit1
```

```
14109   14114   14119   14129   14130   14134   14144   14149   14151   14172
Which type of sorting procedure do you want?
1.quick sort 2.merge sort 3.heap sort 4.exit
Enter choice:2
Time required=3.5e-05 seconds
Which type of sorting procedure do you want?
1.quick sort 2.merge sort 3.heap sort 4.exit
Enter choice:3
Time required=4e-06 seconds
Which type of sorting procedure do you want?
1.quick sort 2.merge sort 3.heap sort 4.exit
Enter choice:4
terminating condition entered
Enter no. of elements in multiple of 10:10
Which type of list you want?
1.ascending 2.descending 3.ascending near sorted 4.random
Enter choice:4
Do you want to copy the generated array? Enter 1 for yes or any other no. to exit:1
Enter 1 if you want to display copied array otherwise 0:1
1678    575     2409    1742    1429    1043    2668    2451    2224    518
Which type of sorting procedure do you want?
1.quick sort 2.merge sort 3.heap sort 4.exit
Enter choice:1
Time required=2e-06 seconds
Do you want to display sorted array?
enter 1 for yes and any other no.to exit1
518     575     1043    1429    1678    1742    2224    2409    2451    2668
Which type of sorting procedure do you want?
1.quick sort 2.merge sort 3.heap sort 4.exit
Enter choice:2
```

```
Time required=3.4e-05 seconds

Which type of sorting procedure do you want?

1.quick sort 2.merge sort 3.heap sort 4.exit

Enter choice:1

Time required=4e-06 seconds

Which type of sorting procedure do you want?

1.quick sort 2.merge sort 3.heap sort 4.exit

Enter choice:4

terminating condition entered
```