Computer Science and Engineering Discipline
Khulna University, Khulna - 9208
Roll#:190208
Name:Tanvir Hassan
email:tanvir1908@cseku.ac.bd
Mobile:01989567104

Assignment on Merge and Quick Sort

Generate random numbers and implement the Merge sort and Quick sort algorithms separately with the same sets of data **using C/C++** (Any other programming language **is not allowed**). Input will be the different numbers of positive integers. Execute your program and fill in the following Table 1 and provide other information.

I. Write the code of your program below.

A. Merge sort:

```
#include<stdio.h>


void mergesort(int a[],int i,int j);

void merge(int a[],int i1,int j1,int i2,int j2);


int data_compare=0;

int data_movement=0;


int main()

{

        int n,i;

        printf("Enter how many data do you want : ");

   scanf("%d",&n);


   int a[n];

   printf("Data : \n");

   for(i=0 ; i<n ; i++)
```

```
    {
        a[i] = rand() % n;
    }

    printf("Before sorting : ");
    for (i=0;i<=n-1;i++)
    {
        printf(" %d", a[i]);
    }
    printf("\n\n\n");

            mergesort(a,0,n-1);

            printf("\nSorted array is :");
            for(i=0;i<n;i++)
                    printf("%d ",a[i]);

                    printf("\n\ndata compared %d times.",data_compare);
                    printf("\n\ndata moved %d times.",data_movement);

            return 0;
}
```

```
void mergesort(int a[],int i,int j)
{
	int mid;

	if(i<j)
	{
		mid=(i+j)/2;
		mergesort(a,i,mid);
		mergesort(a,mid+1,j);
		merge(a,i,mid,mid+1,j);
	}
}

void merge(int a[],int i1,int j1,int i2,int j2)
{
	int temp[j2];
	int i,j,k;
	i=i1;
	j=i2;
	k=0;
	j1;
	j2;

	while(i<=j1 && j<=j2)
```

```
        {
            data_compare++;
            data_movement++;
                if(a[i]<a[j])
                        temp[k++]=a[i++];
                else
                        temp[k++]=a[j++];
        }

        while(i<=j1)
                temp[k++]=a[i++];
                data_movement++;

        while(j<=j2)
                temp[k++]=a[j++];
                data_movement++;

        for(i=i1,j=0;i<=j2;i++,j++)
                a[i]=temp[j];
}
```

B. Quick sort:

```
#include <stdio.h>
void quicksort_method (long long int [], long long int, long long int);
long long int data_movement = 0;
long long int data_compare = 0;
```

```
long long int main()
{
long long int n,i;



        printf("Enter how many data do you want : ");
    scanf("%lld",&n);


    long long int a[n];
    printf("Data : \n");
    for(i=0 ; i<n ; i++)
    {
        a[i] = rand() % n ;
    }


    printf("Before sorting : ");
    for (i=0;i<=n-1;i++)
    {
        printf(" %lld", a[i]);
    }
    printf("\n\n\n");




quicksort_method(a, 0, n - 1);
```

```
printf("\nSorted array is :");
        for(i=0;i<n;i++)
                printf("%lld ",a[i]);


printf("\n");

printf("\nData moved %d times",data_movement);
printf("\nData compared %d times",data_compare);

return 0;
}


void quicksort_method(long long int a[],long long int low,long long int high)
{
long long int pivot, value1, value2, temp;
if (low < high)
{
pivot = low;
value1 = low;
value2 = high;
```

```
data_compare++;
while (value1 < value2)
{
    data_movement++;

while (a[value1] <= a[pivot] && value1 <= high)
{
value1++;
}
while (a[value2] > a[pivot] && value2 >= low)
{
value2--;
}
if (value1 < value2)
{
temp = a[value1];
a[value1] = a[value2];
a[value2] = temp;
}

}
temp = a[value2];
a[value2] = a[pivot];
a[pivot] = temp;
quicksort_method(a, low, value2 - 1);
```

```
quicksort_method(a, value2 + 1, high);

}

}
```

II. Fill in Table 1.

Table 1: Simulation results

No. of Integers	Sorting Algorithm	No. of data comparisons	No. of data movements	Execution time (sec)	Difference in execution time	Comments (if any)
200	Merge	1281	1679	3.477	1.365	
	Quick	133	376	2.112		
500	Merge	3852	4850	2.916	0.403	
	Quick	332	1086	2.513		
1000	Merge	8715	10713	2.585	0.038	
	Quick	689	2329	2.547		
10000	Merge	120425	140423	4.362	0.97	
	Quick	6818	31421	3.392		
1000000	Merge	1536329	1736327			Used web compiller
	Quick	67751	389928			
10000000	Merge	18673448	20673446			Used web compiller
	Quick	677328	4688187			

III. Input and Output of the program

1. Input of 200 integers.
2. Output of 200 integers.

A. Merge sort:

Before sorting : 41 67 134 100 169 124 78 158 162 64 105 145 81 27 161 91 195 142 27 36 191 4 102 153 92 182 21 116 118 95 47 126 171 138 69 112 67 99 35 94 103 11 122 133 73 64 141 111 53 68 147 44 62 157 37 59 123 141 129 178 116 35 190 42 88 106 40 142 64 48 46 5 90 129 170 150 6 101 193 148 29 23 84 154 156 40 166 176 131 108 144 39 26 123 137 138 118 82 129 141 33 115 39 58 104 130 177 106 73 186 21 145 124 72 70 29 177 173 97 112 186 90 161 36 155 167 55 174 31 52 150 150 141 124 166 30 107 191 7 137 57 87 153 183 145 109 9 158 21 188 22 146 106 30 13 168 100 191 162 55 10 159 24 137 148 83 195 41 2 150 91 36 174 20 196 21 148 199 68 84 81 134 53 199 18 138 100 188 127 67 128 93 48 83 7 21 110 17 13 114

Sorted array is :2 4 5 6 7 7 9 10 11 13 13 17 18 20 21 21 21 21 21 22 23 24 26 27 27 29 29 30 30 31 33 35 35 36 36 36 37 39 39 40 40 41 41 42 44 46 47 48 48 52 53 53 55 55 57 58 59 62 64 64 64 67 67 67 68 68 69 70 72 73 73 78 81 81 82 83 83 84 84 87 88 90 90 91 91 92 93 94 95 97 99 100 100 100 101 102 103 104 105 106 106 106 107 108 109 110 111 112 112 114 115 116 116 118 118 122 123 123 124 124 124 126 127 128 129 129 129 130 131 133 134 134 137 137 137 138

138 138 141 141 141 141 142 142 144 145 145 145 146 147 148 148 148 150 150 150 150 153 153 154 155 156 157 158 158 159 161 161 162 162 166 166 167 168 169 170 171 173 174 174 176 177 177 178 182 183 186 186 188 188 190 191 191 191 193 195 195 196 199 199

B. Quick sort:

Before sorting : 41 67 134 100 169 124 78 158 162 64 105 145 81 27 161 91 195 142 27 36 191 4 102 153 92 182 21 116 118 95 47 126 171 138 69 112 67 99 35 94 103 11 122 133 73 64 141 111 53 68 147 44 62 157 37 59 123 141 129 178 116 35 190 42 88 106 40 142 64 48 46 5 90 129 170 150 6 101 193 148 29 23 84 154 156 40 166 176 131 108 144 39 26 123 137 138 118 82 129 141 33 115 39 58 104 130 177 106 73 186 21 145 124 72 70 29 177 173 97 112 186 90 161 36 155 167 55 174 31 52 150 150 141 124 166 30 107 191 7 137 57 87 153 183 145 109 9 158 21 188 22 146 106 30 13 168 100 191 162 55 10 159 24 137 148 83 195 41 2 150 91 36 174 20 196 21 148 199 68 84 81 134 53 199 18 138 100 188 127 67 128 93 48 83 7 21 110 17 13 114

Sorted array is :2 4 5 6 7 7 9 10 11 13 13 17 18 20 21 21 21 21 21 22 23 24 26 27 27 29 29 30 30 31 33 35 35 36 36 36 37 39 39 40 40 41 41 42 44 46 47 48 48 52 53 53 55 55 57 58 59 62 64 64 64 67 67 67 68 68 69 70 72 73 73 78 81 81 82 83 83 84 84 87 88 90 90 91 91 92 93 94 95 97 99 100 100 100 101 102 103 104 105 106 106 106 107 108 109 110 111 112 112 114 115 116 116 118 118 122 123 123 124 124 124 126 127 128 129 129 129 130 131 133 134 134 137 137 137 138 138 138 141 141 141 141 142 142 144 145 145 145 146 147 148 148 148 150 150 150 150 153 153 154 155 156 157 158 158 159 161 161 162 162 166 166 167 168 169 170 171 173 174 174 176 177 177 178 182 183 186 186 188 188 190 191 191 191 193 195 195 196 199 199