Practical - 3

Write user defined functions for the following sorting methods and compare their performance by time measurement with random data and Sorted data.

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

Code(Using Selection Sort):

```
#include<stdio.h>
#include<time.h>
 void selection_sort(int[],int);
 void main() {
 int a[10000],n,i;
 time_t;
 double time_taken;
clock_t start,end;
 printf("21012021003_AMIT GOSWAMI\n");
printf("Enter number of elements: ");
scanf("%d",&n);
 srand((unsigned) time(&t));
for(i=0;i<n;i++)
 {
 a[i]=rand() % 1000;
 }  start=clock();
 selection_sort(a,n);
end=clock();
printf("\nSorted randomly generated elements are: ");
```

```
 for(i=0;i<n;i++)
{
printf("%d",a[i]);
 }
 time_taken=(double)(end-start)/CLOCKS_PER_SEC;
printf("\nTime taken by selection sort for random unsorted array: %lf",time_taken);
 for(i=0;i<n;i++) {
  a[i]=i+1;
 }
  start=clock();
selection_sort(a,n);
end=clock();
printf("\n\nSorted elements are: ");
```

```
for(i=0;i<n;i++) {
 printf("%d ",a[i]);
 }
time_taken=(double)(end-start)/CLOCKS_PER_SEC;
printf("\nTime taken by selection sort for sorted array%lf",time_taken);
}
void selection_sort(int a[],int
n)
{
 int i,j,min,temp;
for(i=0;i<n-1;i++)
 {
 min=i;  for(j=i+1;j<n;j++)
 {
if(a[min]>a[j])
 {
min=j;
 }
 }
 if(min!=i)  {
temp=a[i];
```

```
a[i]=a[min];
a[min]=temp;
 }
 }
}
```

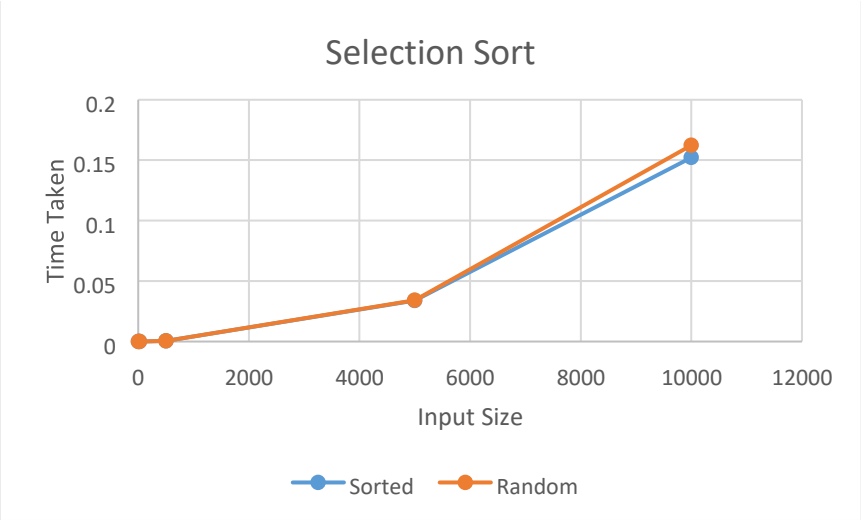
Output:

```
21012021003_AMIT GOSWAMI
Enter number of elements: 5
Sorted randomly generated elements are: 355 525 738 776 997
Time taken by selection sort for random unsorted array: 0.000002

Sorted elements are: 1 2 3 4 5
Time taken by selection sort for sorted array: 0.000001
```

Analysis:

Input Size	Sorted	Random
5	0.000001	0.000002
25	0.000003	0.000005
500	0.000425	0.000433
5000	0.033747	0.034069
10000	0.152171	0.162309



Code(Using Bubble Sort):

```
#include<stdio.h>
#include<time.h>
void bubble_sort(int [],int);
void main()
{
int a[10000],n,i;
time_t t;
double time_taken;
clock_t start,end;
 printf("21012021003_AMIT GOSWAMI \n");
printf("Enter number of elements: ");
scanf("%d",&n);  srand((unsigned) time(&t));
for(i=0;i<n;i++)
 {
 a[i]=rand() %1000;
 }
 start=clock();  bubble_sort(a,n);
 end=clock();
printf("\nSorted randomly generated elements are: ");
for(i=0;i<n;i++)
{
 printf("%d ",a[i]);
 }
 time_taken=(double)(end-start)/CLOCKS_PER_SEC;
printf("\nTime taken by bubble sort for random unsorted array: %lf",time_taken);
for(i=0;i<n;i++)
 {
a[i]=i+1;
 }
```

```
 start=clock();
 bubble_sort(a,n);
 end=clock();
 printf("\n\nSorted elements are:");
for(i=0;i<n;i++)
 {
printf("%d ",a[i]);
 }
 time_taken=(double)(end-start)/CLOCKS_PER_SEC;
 printf("\nTime taken by bubble sort for sorted array: %lf",time_taken);
}
void bubble_sort(int a[],int n)
{
 int i,j,temp,flag;
for(i=0;i<n-1;i++)
 {
flag=0;
for(j=0;j<n-1-i;j++)
 {
if(a[j]>a[j+1])
 {
  temp=a[j];
a[j]=a[j+1];
a[j+1]=temp;
flag=1;
 }
}
if(flag==0)
 {
break;
 }
 }
}
```

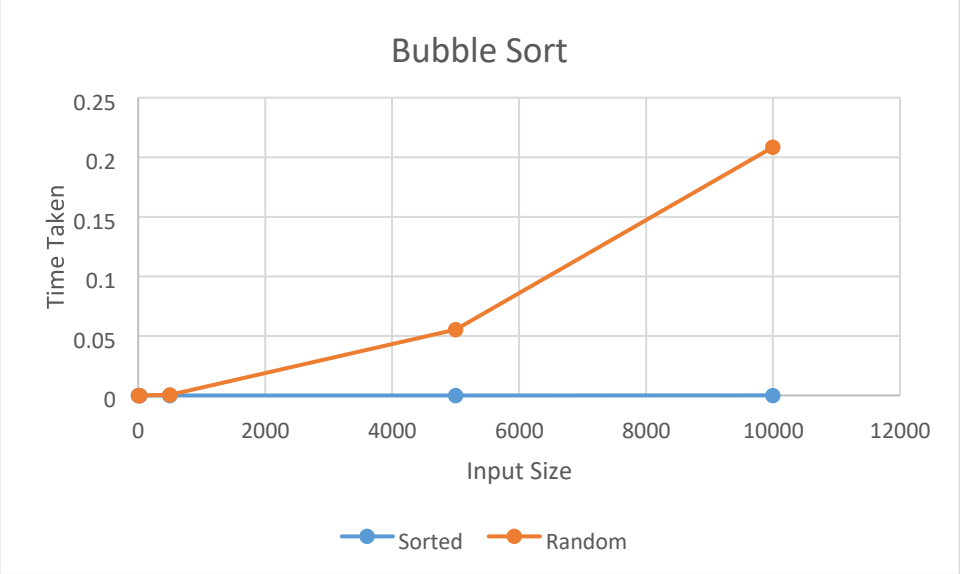
Output:

```
21012021003_AMIT GOSWAMI
Enter number of elements: 5
Sorted randomly generated elements are: 139 274 391 597 916
Time taken by bubble sort for random unsorted array: 0.000003

Sorted elements are: 1 2 3 4 5
Time taken by bubble sort for sorted array: 0.000002
```

Analysis:

Input Size	Sorted	Random
5	0.000002	0.000003
25	0.000002	0.000005
500	0.000005	0.000581
5000	0.000016	0.055213
10000	0.000035	0.208471



Code(Using Insertion Sort):

```
#include<stdio.h>
#include<time.h>
void insertion_sort(int [],int);
void main()
{
int a[10000],n,i;
time_t t;
double time_taken;
clock_t start,end;
 printf("21012021003_AMIT GOSWAMI \n");
printf("Enter number of elements: ");
scanf("%d",&n);
  srand((unsigned) time(&t));
for(i=0;i<n;i++)
 {
a[i]=rand() % 1000;
 }
start=clock();
 insertion_sort(a,n);
 end=clock();
printf("\nSorted randomly generated elements are: ");
for(i=0;i<n;i++)
 {
 printf("%d ",a[i]);
 }
 time_taken=(double)(end-start)/CLOCKS_PER_SEC;
printf("\nTime taken by insertion sort for random unsorted array: %lf",time_taken);
for(i=0;i<n;i++)
 {
a[i]=i+1;
 }
```

```
 start=clock();
 insertion_sort(a,n);
end=clock();
printf("\n\nSorted elements are: ");
for(i=0;i<n;i++)
{
printf("%d ",a[i]);
 }
 time_taken=(double)(end-start)/CLOCKS_PER_SEC;
printf("\nTime taken by insertion sort for sorted array: %lf",time_taken);
}
void insertion_sort(int a[],int
n)
{
 int i,j,temp;
for(i=1;i<n;i++)
 {
temp=a[i];
j=i-1;
 while(j>=0 && a[j]>temp)
 {
a[j+1]=a[j];
 j--;
 }
 a[j+1]=temp;
 }
}
```

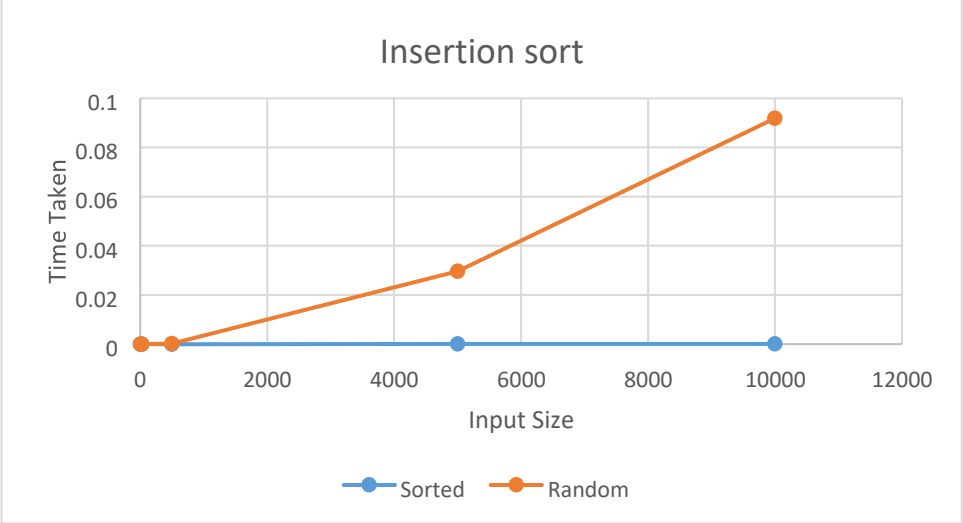
Output:

```
21012021003_AMIT GOSWAMI
Enter number of elements: 5
Sorted randomly generated elements are: 373 439 801 813 954
Time taken by insertion sort for random unsorted array: 0.000003

Sorted elements are: 1 2 3 4 5
Time taken by insertion sort for sorted array: 0.000002
```

Analysis:

Input Size	Sorted	Random
5	0.000002	0.000003
25	0.000001	0.000003
500	0.000004	0.000205
5000	0.000034	0.029683
10000	0.000067	0.091895



Code(Using Merge Sort):

```
#include<stdio.h>
#include<time.h>
 int a[10000],b[10000];
void merge(int a[],int lb,int mid,int ub)
{
int i,j,k; i=lb; j=mid+1;
k=lb;
while(i<=mid && j<=ub)
{
 if(a[i]<=a[j])
{
b[k]=a[i];
i++;
 k++;
}
 else{
b[k]=a[j];
j++;
k++;
}
}
if(i>mid)
 {
while(j<=ub)
{
 b[k]=a[j];
j++;
 k++;
}
}
else{
```

```
while(i<=mid)
{
 b[k]=a[i];
i++;
 k++;
} } for(i=lb;i<=ub;i++)
{ a[i]=b[i];
}
 }
void merge_sort(int a[],int lb,int ub)
{
int mid;
if(lb<ub) {
 mid=(lb+ub)/2;
merge_sort(a,lb,mid);
merge_sort(a,mid+1,ub);
merge(a,lb,mid,ub);
}
 }
 void
main() {
int n,i,j;
time_t t;
double
time_taken;
clock_t start,end;
 printf("21012021003_AMIT GOSWAMI \n");
printf("Enter number of elements: ");
scanf("%d",&n);
srand((unsigned) time(&t)); for(i=0;i<n;i++) {
a[i]=rand() % 1000;
}
```

```
 start=clock();
merge_sort(a,0,n-1);
end=clock();
printf("\nSorted randomly generated elements are: ");
for(i=0;i<n;i++)
{
printf("%d",a[i]);
}
time_taken=(double)(end-start)/CLOCKS_PER_SEC;
printf("\nTime taken by merge sort for random unsorted array:%lf",time_taken);
for(i=0;i<n;i++)
{
a[i]=i+1;
} start=clock();
merge_sort(a,0,n-1);
 end=clock();
printf("\n\nSorted elements are:");
for(i=0;i<n;i++)
{
printf("%d",a[i]);
}
time_taken=(double)(end-start)/CLOCKS_PER_SEC;
 printf("\nTime taken by merge sort for sorted array%lf",time_taken);
}
```

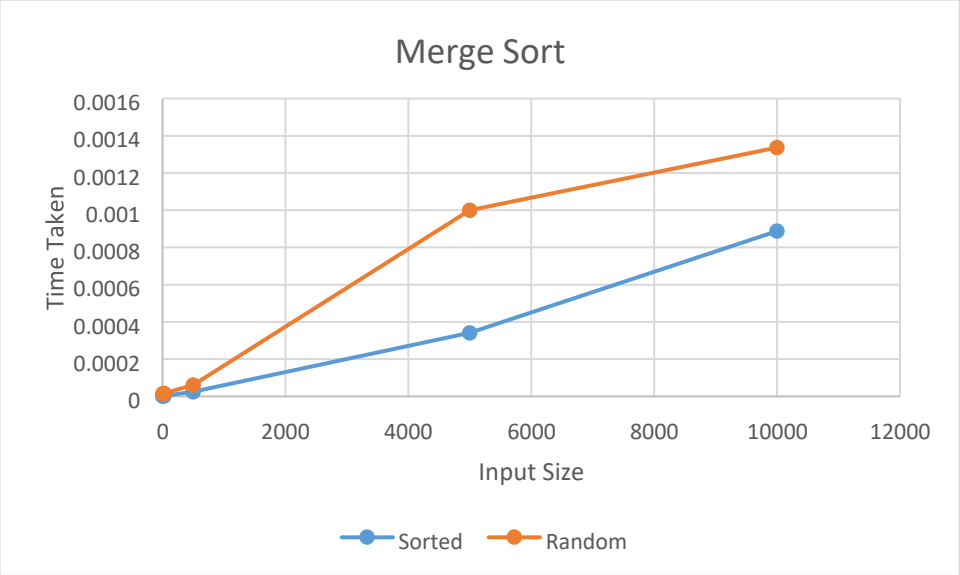
Output:

```
Enter number of elements: 5
Sorted randomly generated elements are: 27 269 797 847 987
Time taken by merge sort for random unsorted array: 0.000013

Sorted elements are: 1 2 3 4 5
Time taken by merge sort for sorted array: 0.000002
```

Analysis:

Input Size	Sorted	Random
5	0.000002	0.000013
25	0.000002	0.000016
500	0.000025	0.000061
5000	0.000342	0.001001
10000	0.000888	0.001337



Code(Using Quick Sort):

```
#include<stdio.h>
#include<time.h>
void quick_sort(int a[],int lb,int ub)
{
int pivot,start,end,temp;
if(lb<ub)
 {
 pivot=lb;
start=lb;
end=ub;
while(start<end)
 {
while(a[pivot]>=a[start] )
 {
start++;
 }
while(a[pivot]<a[end])
 {
 end--;
  }
if(start<end)
 {
 temp=a[start];
a[start]=a[end];
a[end]=temp;
 }
 }
  temp=a[pivot];
a[pivot]=a[end];
a[end]=temp;


 quick_sort(a,lb,end-1);
 quick_sort(a,end+1,ub);
 }
```

```
}
 int main() {
 int a[10000],i,n;
time_t t;
 double
time_taken;
clock_t start,end;
 printf("21012021003_AMIT GOSWAMI \n");
printf("Enter number of elements: ");
scanf("%d",&n);
 srand((unsigned) time(&t));  for(i=0;i<n;i++)
 {
a[i]=rand()%1000;
 }
 start=clock();
 quick_sort(a,0,n-1);
end=clock();
printf("\nSorted randomly generated elements are");
  for(i=0;i<n;i++)
{
```

```
printf("%d",a[i]);
 }
 time_taken=(double)(end-start)/CLOCKS_PER_SEC;
 printf("\nTime taken by quick sort for random unsorted array: %lf",time_taken);
for(i=0;i<n;i++)
 {
a[i]=i+1;
}
 start=clock();
quick_sort(a,0,n-1);
end=clock();
 printf("\n\nSorted elements are: ");
 for(i=0;i<n;i++)
 {
 printf("%d
",a[i]);
 }
 time_taken=(double)(end-start)/CLOCKS_PER_SEC;
printf("\nTime taken by quick sort for sorted array: %lf",time_taken);
return 0;
}
```

Output:

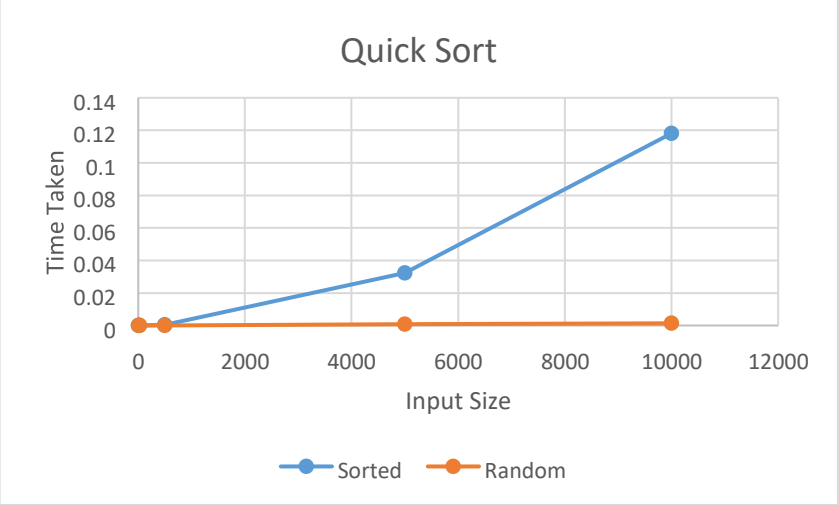
```
21012021003_AMIT GOSWAMI
Enter number of elements: 5
Sorted randomly generated elements are: 19 237 570 808 925
Time taken by quick sort for random unsorted array: 0.000002

Sorted elements are: 1 2 3 4 5
Time taken by quick sort for sorted array: 0.000002
```

Analysis:

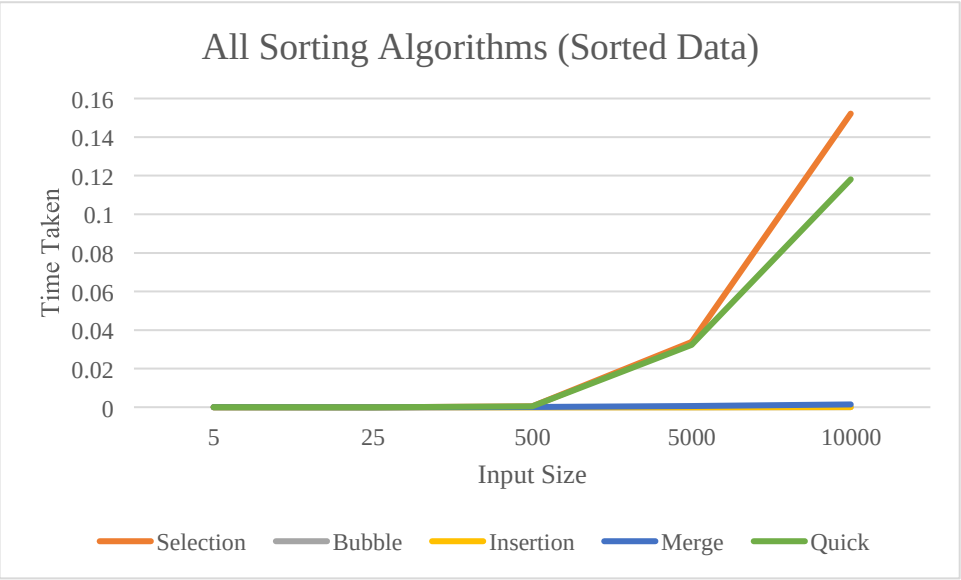
Input Size	Sorted	Random
5	0.000002	0.00002
25	0.000003	0.000005
500	0.000397	0.000107

5000	0.032309	0.000820
10000	0.118116	0.001367



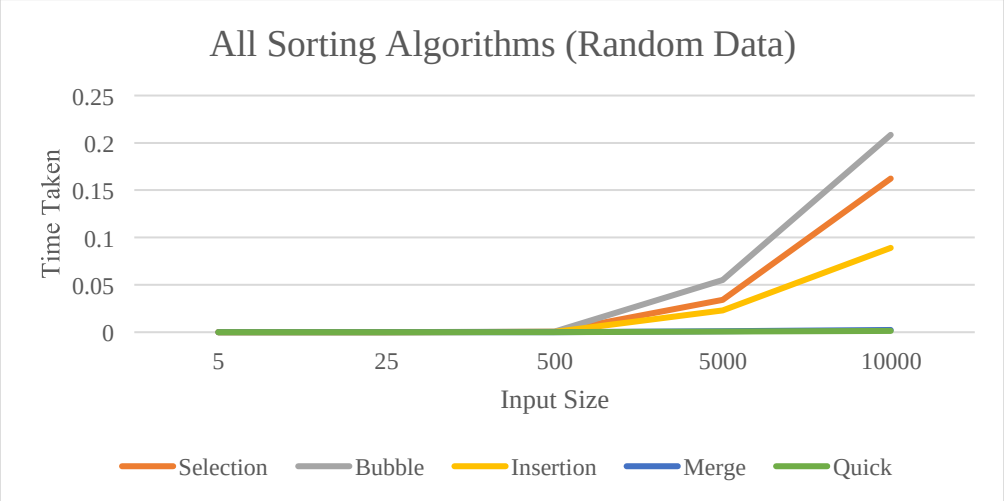
Analysis of all sorting types (Sorted Data):

Input Size	Selection Sort	Bubble Sort	Insertion Sort	Merge Sort	Quick Sort
5	0.000002	0.000001	0.000002	0.000001	0.000002
25	0.000003	0.000002	0.000001	0.000002	0.000003
500	0.000425	0.000005	0.000004	0.000025	0.000397
5000	0.033747	0.000016	0.000034	0.000342	0.032309
10000	0.152171	0.000035	0.000067	0.000888	0.118116



Analysis of all sorting types (Random Data):

Input Size	Selection Sort	Bubble Sort	Insertion Sort	Merge Sort	Quick Sort
5	0.000002	0.000003	0.000003	0.000013	0.00002
25	0.000005	0.000005	0.000003	0.000016	0.000005
500	0.000433	0.000581	0.000205	0.000061	0.000107
5000	0.034069	0.055213	0.029683	0.001001	0.000820
10000	0.162309	0.208471	0.091895	0.001337	0.001367



Conclusion:

Various sorting algorithms perform differently for sorted and unsorted data. Thus it is advisable to choose correct algorithms according to user's needs.