INDEX

NAME:-SHUBHAM KAPOOR **E-ROLL : 0131CS161108**

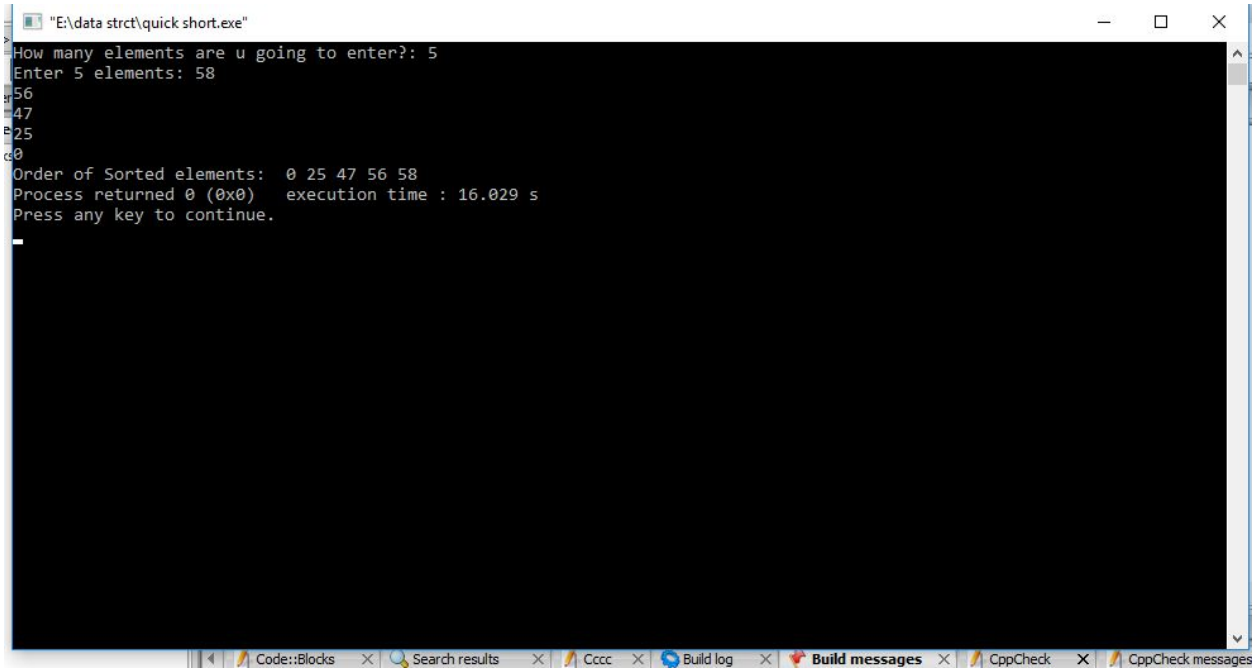
SUBJECT:- DATA STRUCTURE **YEAR:- 2nd year (3 sem)**

S.NO.	EXPERIMENT	DATE EXP.	SUBMISSION	SING.
1.	INSERTION SORT			
2.	SELECTION SORT			
3.	QUICK SORT			
4.	SHELL SORT			
5.	BUCKET SORT			
6.	MERGE SORT			
7.	MAX HEAP			
8.	MIN HEAP			
9.	RADIX SORT			

QUICK SORT

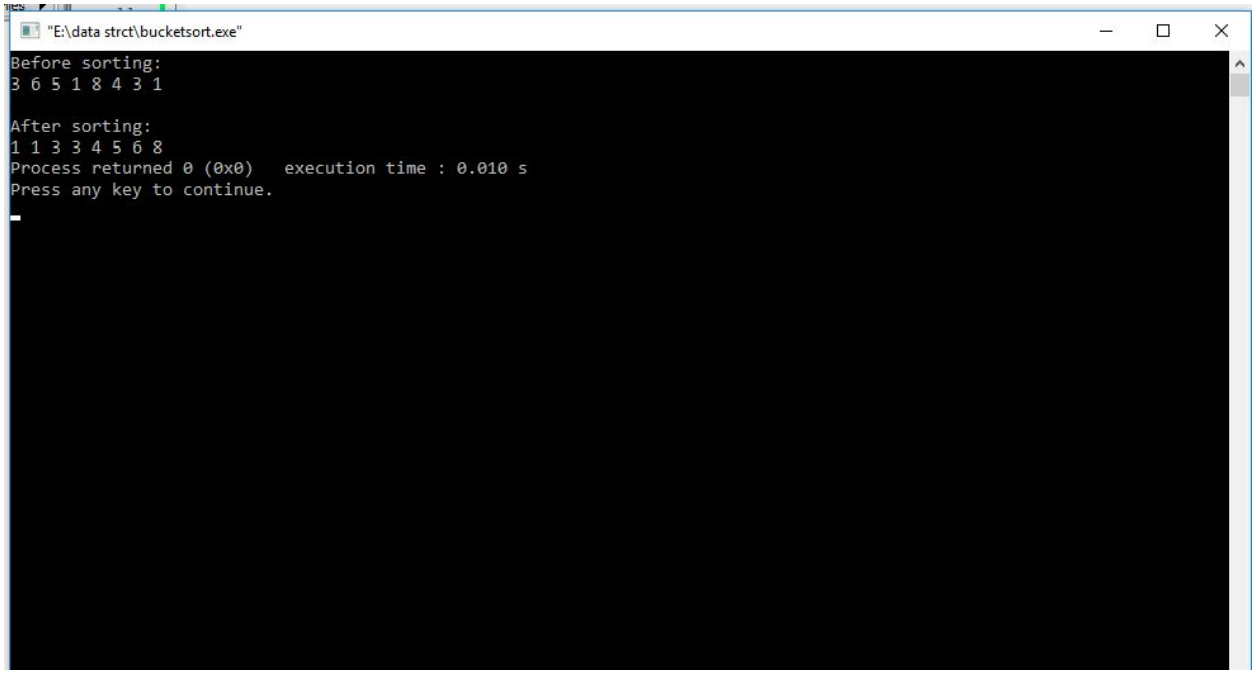
```
#include<stdio.h>
void quicksort(int number[25],int first,int last){
  int i, j, pivot, temp;

  if(first<last){
    pivot=first;
    i=first;
    j=last;

    while(i<j){
      while(number[i]<=number[pivot]&&i<last)
        i++;
      while(number[j]>number[pivot])
        j--;
      if(i<j){
        temp=number[i];
        number[i]=number[j];
        number[j]=temp;
      }
    }

    temp=number[pivot];
    number[pivot]=number[j];
    number[j]=temp;
    quicksort(number,first,j-1);
    quicksort(number,j+1,last);

  }
}
```

```
int main(){
   int i, count, number[25];
  printf("How many elements are u going to enter?: ");
   scanf("%d",&count);
   printf("Enter %d elements: ", count);
   for(i=0;i<count;i++)
  scanf("%d",&number[i]);
 quicksort(number,0,count-1);
 printf("Order of Sorted elements: ");
   for(i=0;i<count;i++)
     printf(" %d",number[i]);

   return 0;
}
```



BUCKET SORT

```
#include<stdio.h>

#define SIZE 10

void bucketSort(int a[], int n) {
    int i, j, k, buckets[SIZE];

    for(i = 0; i < SIZE; ++i)
        buckets[i] = 0;

    for(i = 0; i < n; ++i)
        ++buckets[a[i]];

    for(i = 0, j = 0; j < SIZE; ++j)
        for(k = buckets[j]; k > 0; --k)
            a[i++] = j;
}

int main() {
    int i, a[] = {3, 6, 5, 1, 8, 4, 3, 1}, n = 8;

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

    bucketSort(a, n);
 printf("\n\nAfter sorting:\n");
    for(i = 0; i < n; ++i)
        printf("%d ", a[i]);
   return 0;}
```


"E:\data strct\bucketsort.exe"
Before sorting:
3 6 5 1 8 4 3 1
After sorting:
1 1 3 3 4 5 6 8
Process returned 0 (0x0) execution time : 0.010 s
Press any key to continue.

MAX HEAP

```
#include <stdio.h>

void main()
{
   int heap[10], no, i, j, c, root, temp;

   printf("\n Enter no of elements :");
   scanf("%d", &no);
   printf("\n Enter the nos : ");
   for (i = 0; i < no; i++)
      scanf("%d", &heap[i]);
   for (i = 1; i < no; i++)
   {
       c = i;
       do
       {
           root = (c - 1) / 2;
           if (heap[root] < heap[c])
           {
               temp = heap[root];
               heap[root] = heap[c];
               heap[c] = temp;
           }
           c = root;
       } while (c != 0);
   }

   printf("Heap array : ");
   for (i = 0; i < no; i++)
       printf("%d\t ", heap[i]);
   for (j = no - 1; j >= 0; j--)
```

```
    {
        temp = heap[0];
        heap[0] = heap[j];
        heap[j] = temp;
        root = 0;
        do
        {
            c = 2 * root + 1;
            if ((heap[c] < heap[c + 1]) && c < j-1)
                c++;
            if (heap[root]<heap[c] && c<j)
            {
                temp = heap[root];
                heap[root] = heap[c];
                heap[c] = temp;
            }
            root = c;
        } while (c < j);
    }
    printf("\n The sorted array is : ");
    for (i = 0; i < no; i++)
       printf("\t %d", heap[i]);

}
```

"E:\data strct\heap max tree.exe"

```
 Enter no of elements :6

 Enter the nos : 45
24
100
147
999
1001
Heap array : 1001        147     999     24      100     45
 The sorted array is :   24      45      100     147     999     1001
Process returned 6 (0x6)   execution time : 32.156 s
Press any key to continue.
```

MIN HEAP

```
#include<stdio.h>
void heapsort(int[],int);
void heapify(int[],int);
void adjust(int[],int);
main() {
        int n,i,a[50];
        system("clear");
        printf("\nEnter the limit:");
        scanf("%d",&n);
        printf("\nEnter the elements:");
        for (i=0;i<n;i++)
          scanf("%d",&a[i]);
        heapsort(a,n);
        printf("\nThe Sorted Elements Are:\n");
        for (i=0;i<n;i++)
          printf("\t%d",a[i]);
        printf("\n");
}
void heapsort(int a[],int n) {
        int i,t;
        heapify(a,n);
        for (i=n-1;i>0;i--) {
                t = a[0];
                a[0] = a[i];
                a[i] = t;
                adjust(a,i);
        }
}
void heapify(int a[],int n) {
        int k,i,j,item;
        for (k=1;k<n;k++) {
```

```
                item = a[k];
                i = k;
                j = (i-1)/2;
                while((i>0)&&(item>a[j])) {
                        a[i] = a[j];
                        i = j;
                        j = (i-1)/2;
                }
                a[i] = item;
        }
}
void adjust(int a[],int n) {
        int i,j,item;
        j = 0;
        item = a[j];
        i = 2*j+1;
        while(i<=n-1) {
                if(i+1 <= n-1)
                   if(a[i] <a[i+1])
                    i++;
                if(item<a[i]) {
                        a[j] = a[i];
                        j = i;
                        i = 2*j+1;
                } else
                   break;
        }
        a[j] = item;
}
```

"E:\data strct\heap max sort.exe"

```
'clear' is not recognized as an internal or external command,
operable program or batch file.

Enter the limit:5

Enter the elements:45
0
25
65
45

The Sorted Elements Are:
        0       25      45      45      65

Process returned 10 (0xA)   execution time : 15.056 s
Press any key to continue.
```

Radix sort

```
#include<stdio.h>


int largest(int a[], int n)
{
    int large = a[0], i;
    for(i = 1; i < n; i++)
    {
        if(large < a[i])
            large = a[i];
    }
    return large;
}


void RadixSort(int a[], int n)
{
    int bucket[10][10], bucket_count[10];
    int i, j, k, remainder, NOP=0, divisor=1, large, pass;

    large = largest(a, n);
    printf("The large element %d\n",large);
    while(large > 0)
    {
        NOP++;
        large/=10;
    }

    for(pass = 0; pass < NOP; pass++)
    {
```

```
        for(i = 0; i < 10; i++)
        {
            bucket_count[i] = 0;
        }
        for(i = 0; i < n; i++)
        {
            remainder = (a[i] / divisor) % 10;
            bucket[remainder][bucket_count[remainder]] = a[i];
            bucket_count[remainder] += 1;
        }

        i = 0;
        for(k = 0; k < 10; k++)
        {
            for(j = 0; j < bucket_count[k]; j++)
            {
                a[i] = bucket[k][j];
                i++;
            }
        }
        divisor *= 10;

        for(i = 0; i < n; i++)
            printf("%d  ",a[i]);
        printf("\n");
    }
}

int main()
{
    int i, n, a[10];
    printf("Enter the number of elements :: ");
    scanf("%d",&n);
```

```
    printf("Enter the elements :: ");
    for(i = 0; i < n; i++)
    {
        scanf("%d",&a[i]);
    }
    RadixSort(a,n);
    printf("The sorted elements are ::  ");
    for(i = 0; i < n; i++)
        printf("%d  ",a[i]);
    printf("\n");
    return 0;
}
```