Date: -10.11.2022

Program: -To perform Selection Sorting.

```
#include<stdio.h>
#define SIZE 10
void selection_sort(int[],int);
int main()
{
   int a[SIZE],n,i;
   printf("enter how many elements:");
   scanf("%d",&n);
   for(i=0;i<n;i++)
   {
      printf("enter element %d:",i+1);
      scanf("%d",&a[i]);
   }
   selection_sort(a,n);
   for(i=0;i<n;i++)
      printf("%d\n",a[i]);
   return 0;
}
void selection_sort(int a[],int n)
{
   int minpos;
   int i,j,t;
   for(i=0;i<n-1;i++)
   {
      minpos=i;
      for(j=i+1;j<n;j++)
      {
         if(a[minpos]>a[j])
         {
```

```
            minpos=j;
        }
    }
    if(minpos!=i)
    {
        t=a[i];
        a[i]=a[minpos];
        a[minpos]=t;
    }
  }
}
```

Output: -

```
enter how many elements:5
enter element 1:2
enter element 2:3
enter element 3:4
enter element 4:1
enter element 5:7
1
2
3
4
7
```

Program: - To perform Insertion Sorting

```
#include<stdio.h>
#define SIZE 10
void insertion_sort(int [],int);
int main()
{
   int a[SIZE],i,n;
   printf("enter how many elements:");
   scanf("%d",&n);
   for(i=0;i<n;i++)
   {
      printf("enter element %d:",i+1);
      scanf("%d",&a[i]);
   }
   insertion_sort(a,n);
   for(i=0;i<n;i++)
   printf("%d\n",a[i]);
   return 0;
}
void insertion_sort(int a[],int n)
{
   int i,j,item;
   for(i=1;i<n;i++)
   {
      item=a[i];
      for(j=i-1;j>=0&&a[j]>item;j--)
         a[j+1]=a[j];
      a[j+1]=item;
   }
}
```

Output: -

```
enter how many elements:5
enter element 1:1
enter element 2:4
enter element 3:2
enter element 4:5
enter element 5:3
1
2
3
4
5
```

Program: - To perform Quick Sorting.

```
#include<stdio.h>
#define SIZE 10
int partition(int [],int,int);
void quick_sort(int [],int,int);
int main()
{
   int a[SIZE],n,i;
   printf("enter how many elements:");
   scanf("%d",&n);
   for(i=0;i<n;i++)
   {
      printf("enter element %d:",i+1);
      scanf("%d",&a[i]);
   }
   quick_sort(a,0,n-1);
   for(i=0;i<n;i++)
      printf("%d\n",a[i]);
   return 0;
}
int partition(int a[],int lb,int ub)
{
   int down,up,t,pivot;
   down=lb;
   up=ub;
   pivot=a[lb];
   while(down<up)
   {
      while(a[down]<=pivot&&down<up)
         down++;
      while(a[up]>pivot)
```

```
        up--;
      if(down<up)
      {
        t=a[down];
        a[down]=a[up];
        a[up]=t;
      }
    }
    a[lb]=a[up];
    a[up]=pivot;
    return(up);
}
void quick_sort(int a[],int lb,int ub)
{
    int mid;
    if(lb>=ub)
      return;
    mid=partition(a,lb,ub);
    quick_sort(a,lb,mid-1);
    quick_sort(a,mid+1,ub);
}
```

Output: -

```
enter how many elements:5
enter element 1:2
enter element 2:4
enter element 3:5
enter element 4:3
enter element 5:1
1
2
3
4
5
```

Program: - To perform Heap Sorting.

```
#include<stdio.h>
#define SIZE 10
void insheap(int[],int,int);
int delheap(int[],int);
void heapsort(int[],int);
int main()
{
	int a[SIZE],i,n;
	printf("enter how many elements:");
	scanf("%d",&n);
	for(i=0;i<n;i++)
	{
		printf("enter element%d:",i+1);
		scanf("%d",&a[i]);
	}
	heapsort(a,n);
	for(i=0;i<n;i++)
		printf("%d\n",a[i]);
	return 0;
}
void insheap(int tree[],int n,int item)
{
	int ptr,par;
	n++;
	ptr=n;
	while(ptr>0)
	{
		par=(ptr-1)/2;
		if(item<=tree[par])
		{
```

```
                tree[ptr]=item;
                return;
            }
            tree[ptr]=tree[par];
            ptr=par;
        }
        tree[0]=item;
}
int delheap(int tree[],int n)
{
        int item,ptr,last,left,right;
        item=tree[0];
        last=tree[n];
        ptr=0;
        left=1;
        right=2;
        while(right<=n)
        {
            if(last>=tree[left]&&last>=tree[right])
            {
                tree[ptr]=last;
                return(item);
            }
            if(tree[right]<=tree[left])
            {
                tree[ptr]=tree[left];
                ptr=left;
            }
            else
            {
                tree[ptr]=tree[right];
```

```
                ptr=right;
            }
            left=2*ptr+1;
            right=left+1;
        }
        if(left==n-1&&last<tree[left])
        {
            tree[ptr]=tree[left];
            ptr=left;
        }
        tree[ptr]=last;
        return(item);
}
void heapsort(int a[],int n)
{
        int item,j;
        for(j=0;j<n-1;j++)
            insheap(a,j,a[j+1]);
        while(n>0)
        {
            item=delheap(a,n-1);
            a[n-1]=item;
            n--;
        }
}
```

Output:-

```
enter how many elements:5
enter element1:3
enter element2:2
enter element3:4
enter element4:1
enter element5:5
1
2
3
4
5
```

Program: -To perform Shell Sorting

```
#include<stdio.h>
#define SIZE 10
void shell_sort(int[],int);
void main()
{
   int a[SIZE],i,n;
   printf("enter how many elements:");
   scanf("%d",&n);
   for(i=0;i<n;i++)
   {
      printf("enter element %d:",i+1);
      scanf("%d",&a[i]);
   }
   shell_sort(a,n);
   for(i=0;i<n;i++)
      printf("%d\n",a[i]);
}
void shell_sort(int a[],int n)
{
   int i,j,item,span;
   span=n/2;
   while(span>=1)
   {
      for(i=span;i<n;i++)
      {
         item=a[i];
         for(j=i-span;j>=0&&a[j]>item;j-=span)
            a[j+span]=a[j];
         a[j+span]=item;
      }
```

```
        span=span/2;
    }
}
```

Output:

```
enter how many elements:5
enter element 1:3
enter element 2:4
enter element 3:5
enter element 4:2
enter element 5:8
2
3
4
5
8
```

Program: - To perform Merge Sorting.

```
#include<stdio.h>
#include<stdlib.h>
void merge(int a[],int ub,int mid,int lb)
{
   int i,j,k;
   int n1=mid-ub+1;
   int n2=lb-mid;
   int leftarray[n1],rightarray[n2];
   for (int i = 0; i < n1; i++)
   leftarray[i] = a[ub + i];
   for (int j = 0; j < n2; j++)
   rightarray[j] = a[mid + 1 + j];
   i = 0,
   j = 0;
   k = ub;
   while (i < n1 && j < n2)
   {
      if(leftarray[i] <= rightarray[j])
      {
         a[k] = leftarray[i];
         i++;
      }
      else
      {
         a[k] = rightarray[j];
         j++;
      }
      k++;
   }
   while (i<n1)
```

```
    {
        a[k] = leftarray[i];
        i++;
        k++;
    }

    while (j<n2)
    {
        a[k] = rightarray[j];
        j++;
        k++;
    }
}
 void mergeSort(int a[], int ub, int lb)
{
    if (ub < lb)
    {
        int mid = (ub + lb) / 2;
        mergeSort(a, ub, mid);
        mergeSort(a, mid + 1, lb);
        merge(a, ub, mid, lb);
    }
}
void printArray(int a[], int n)
{
    int i;
    for (i = 0; i < n; i++)
        printf("%d ", a[i]);
    printf("\n");
}
int main()
```

```
{
    int a[] = { 12, 31, 25, 8, 32, 17, 40, 42 };
    int n = sizeof(a) / sizeof(a[0]);
    printf("Before sorting array elements are - \n");
    printArray(a, n);
    mergeSort(a, 0, n - 1);
    printf("After sorting array elements are - \n");
    printArray(a, n);
    return 0;
}
```

Output: -

```
Before sorting array elements are -
12 31 25 8 32 17 40 42
After sorting array elements are -
8 12 17 25 31 32 40 42
```

PRACTICAL FILE

ON

Data Structure Using C

[CSIT124]

SUBMITTED TO:

Ms. Neetu Narayan

ASSOCIATE PROFESSOR

SUBMITTED BY:

Name: Ashish Gupta

Enrolment No.: A2305221299

B.Tech. 3CSE6X

COMPUTER SCIENCE AND ENGINEERING

AMITY SCHOOL OF ENGINEERING &TECHNOLOGY

AMITY UNIVERSITY, UTTAR PRADESH

SESSION- 2022-2023