(DATA STRUCTURES AND ALGORITHMS)

(RECURSSION PROGRAMS)

(ALI AKBER BSCS 3RD SS1)

FACTORIAL OF NUMBER.

```
//FACTORIAL
#include <iostream>
using namespace std;
int factorial(int);


int main(){
    int n;
    cout<<"Enter any number"<<endl;
    cin>>n;
    cout<<"Factorial of "<<n<<" is "<<factorial(n)<<endl;
      return 0;
}
  int factorial(int no){
      if(no==0){
          return 1;
      } else{
            return no*factorial(no-1);
      }
    }
```

BASE AND EXPONENT.

```
//BASE AND EXPONENT
```

```
#include <iostream>
using namespace std;
int power(int,int);

int main(){
    int base,exponent;
    cout<<"Enter base"<<endl;
    cin>>base;
      cout<<"Enter exponent"<<endl;
    cin>>exponent;
    cout<<"Result is "<<power(base,exponent)<<endl;
      return 0;
}
  int power(int b,int ex){
      if(ex==0){
          return 1;
      } else{
            return b*power(b,ex-1);
      }
    }
```

BINARY SEARCH

```
//BINARY SEARCH (RECURISION)
#include <iostream>
using namespace std;
int search(int arr[], int lower, int upper, int val);
```

```
int main(){
    const int MAX=10;
    int arr[MAX]={22,51,65,74,77,82,84,87,89,97};
  int r=search(arr,0,MAX-1,65);
  if(r!=-1){
    cout<<"Number found at index "<<r<<endl;
  } else{
    cout<<"Number not found"<<endl;
  }


      return 0;
}
  int search(int arr[],int lower,int upper,int val){
    int mid=(lower+upper)/2;
          if(lower>upper){
            return -1;
          }
            else if(arr[mid]==val){
                return mid;
          }
          else if(arr[mid]<val){
                return search(arr,mid+1,upper,val);
            }
            else{
                return search(arr,lower,mid-1,val);
            }
```

```
        }
```

TOWER OF HANOI

```
// TOWER OF HANOI.
#include <iostream>
using namespace std;
void tower(int,char,char,char);
int main(){
  int no;
  cout<<"Enter number of discs: "<<endl;
  cin>>no;
  tower(no,'A','B','C');
      return 0;
}
void tower(int n,char src,char aux,char des){
    if(n>1){
      tower(n-1,src,des,aux);
    tower(1,src,aux,des);
      tower(n-1,aux,src,des);
    } else{
        cout<<" Move Disc from "<<src<<" to "<<des<<endl;
    }
}
```

QUICK SORT

```
// QUICK SORT.
#include <iostream>
```

```
using namespace std;
const int MAX=20;
class QuickSort{
    private:
    int arr[MAX];
    public:
    QuickSort(){
        arr[0]=3;
        arr[1]=8;
        arr[2]=2;
        arr[3]=56;
        arr[4]=25;
        arr[5]=39;
        arr[6]=36;
    }
    void display(){
        for(int i=0;i<7;i++){
            cout<<arr[i]<<"\t";
            cout<<endl;
        }
    }
    void qsort(int left,int right){
        if(right-left<=0)
        return;
        int pivot=arr[right];
        int partition=partitionit(left,right,pivot);
```

```
        qsort(left,partition-1);
        qsort(partition+1,right);
    }
    int partitionit(int left,int right,int pivot){
        int lb=left-1;
        int ub=right;
        while(true){
            while(arr[++lb]<pivot);
            while(ub>0 && arr[--ub]>pivot);
            if(lb>=ub)
            break;
            else{
                swap(lb,ub);
            }
        }
        swap(lb,right);
        return lb;
    }
    void swap(int no1,int no2){
        int temp=arr[no1];
        arr[no1]=arr[no2];
        arr[no2]=temp;
    }
};

int main(){
```

```
    QuickSort qs;
    qs.display();
    qs.qsort(0,6);
    cout<<"Sorted array is : "<<endl;
    qs.display();

     return 0;
}
```

MERGE SORT

```
//MERGE SORT.
#include <iostream>
#include <string>
using namespace std;
const int MAX=8;
class Merge{
        private:
        int arr[MAX];
        public:
          Merge(){
        arr[0]=6;
        arr[1]=7;
        arr[2]=2;
        arr[3]=5;
        arr[4]=-1;
        arr[5]=3;
        arr[6]=12;
```

```
        arr[7]=8;
    }

    void display(){
        for(int i=0;i<MAX;i++){
            cout<<arr[i]<<"\t";
        }
            cout<<endl;
    }

void sorting(){
int aux[MAX];
recms(aux,0,7);
}

void recms(int aux[],int lower,int upper){
        if(lower==upper)
        return;
        int mid=(lower+upper)/2;
        recms(aux,lower,mid);
          recms(aux,mid+1,upper);
          merge(aux,lower,mid+1,upper);
}

void merge(int aux[],int low,int high,int ub){
        int j=0;
```

```
        int lb=low;
        int mid=high-1;
        int no=ub-lb+1;
while(low<=mid && high<=ub){
        if(arr[low]<arr[high])
        aux[j++]=arr[low++];
        else
          aux[j++]=arr[high++];
}
          while(low<=mid)
                aux[j++]=arr[low++];
                  while(high<=ub)
                aux[j++]=arr[high++];
        for(j=0;j<no;j++)
        arr[lb+j]=aux[j];
}
};

int main() {
        Merge ms;
        cout<<"Given array is "<<endl;
        ms.display();
      ms.sorting();
cout<<"Sorted Array is "<<endl;
    ms.display();
        return 0;
```

```
}
```