LAB 1.

1. AIM: Find Greatest Common Divisor using Euclid's method.

```
#include<stdio.h>
#include<unistd.h>
int main()
{

      int m=0,n=0,t=0;

      printf("Enter the no1:\n");
      scanf("%d",&m);

      printf("Enter the no2:\n");
      scanf("%d",&n);
      t = euclid(m,n);
      printf("GCD of %d & %d is: %d\n",m , n , t);
      return 0;
}
int euclid(int m,int n)

{
      int t;
      while(n!=0)
      {
            t = m % n;
            m = n;
            n = t;
      }
      return m;
}
```

Input output:

```
[user1@centos LAB1]$ ./gcdEuclid.out

Enter the no1:

5

Enter the no2:

10

GCD of 5 & 10 is: 5

[user1@centos LAB1]$ ./gcdEuclid.out
```

```
Enter the no1:

5

Enter the no2:

3

GCD of 5 & 3 is: 1

[user1@centos LAB1]$ ./gcdEuclid.out

Enter the no1:

5

Enter the no2:

5

GCD of 5 & 5 is: 5

[user1@centos LAB1]$
```

Complexity:

2.AIM: Construct Euclid's Game

```
class EuclidGame
{
      public static void main(String args[])
      {
            int m = Integer.parseInt(args[0]);
            int n = Integer.parseInt(args[1]);
            int temp;
            int gcd = 0;
            if(m>n)
            {
                  temp = m;
                  m = n;
                  n = temp;
            }
            gcd = euclid(n,m);

            int no_of_moves = n / gcd;

            if(no_of_moves%2 != 0)//odd no. of runs
```

```
                System.out.println("player 1 wins the game");
            else
                System.out.println("player 2 wins the game");
        }
        static int euclid(int m,int n)
        {
            int t;
            while(n!=0)
            {
                t = m % n;
                m = n;
                n = t;
            }
            return m;
        }
}
```

Input output:

```
[user1@centos LAB1]$ javac EuclidGame.java

[user1@centos LAB1]$ java EuclidGame 5 14

player 2 wins the game

[user1@centos LAB1]$ java EuclidGame 7 10

player 2 wins the game

[user1@centos LAB1]$ java EuclidGame 0 2

player 1 wins the game

[user1@centos LAB1]$ java EuclidGame 1 8

player 2 wins the game

[user1@centos LAB1]$ java EuclidGame 6 3

player 2 wins the game
```

Complexity:

---

3. AIM: Finding Greatest Common Divisor using min method

```
#include<stdio.h>
#include<unistd.h>
```

```
int main()
{
      int m=0,n=0,t=0;

      printf("Enter the no1:\n");
      scanf("%d",&m);

      printf("Enter the no2:\n");
      scanf("%d",&n);

      if(m <= n)
      {
            t = m;
      }
      else
      {
            t = n;
      }
      while(m%t != 0)
      {
            t = t-1;
      }
      prev:
      if(n%t == 0)
      {
            printf("GCD of %d & %d is: %d\n",m , n , t);
            _exit(0);
      }
      t--;
      goto prev;


      return 0;
}
```

Input Output:

```
[user1@centos LAB1]$ ./gcdmin.out

Enter the no1:

5

Enter the no2:

5

GCD of 5 & 5 is: 5

[user1@centos LAB1]$ ./gcdmin.out

Enter the no1:
```

```
5

Enter the no2:

10

GCD of 5 & 10 is: 5

[user1@centos LAB1]$ ./gcdmin.out

Enter the no1:

5

Enter the no2:

3

GCD of 5 & 3 is: 1

[user1@centos LAB1]$
```

Complexity:

4.Find Greatest Common Divisor using Prime Factors

```
#include<stdio.h>
#include<unistd.h>
#include<math.h>
#include<stdlib.h>

#define MAX 100

int mfact[MAX],nfact[MAX],count=1,mfactcnt=0,nfactcnt=0;

int chkPrime(int i)
{
      int j,flag=0;
      for(j=2;j<=i;j++)
      {
            if(i%j == 0)
                  {flag = 1; break;}
            else
                  flag = 0;
      }
      if(flag == 0)
      {
            return 0;
      }
      return 1;
}
```

```
void findprime(int *arr,int no)
{
      int i,j,flag=0,k=0,no1=no;
      for(i=0;i<=MAX;i++)
      {
            arr[i] = 0;
      }
      for(i=2;i<=no || chkPrime(no);)
      {
            if(no%i == 0)
            {
                  no = no / i;
                  if(chkPrime(i))
                  {
                        arr[k] = i;
                        k++;
                        continue;
                  }
            }
            else
            {
                  i++;
                  continue;
            }
      }

      printf("%d's factor are:\n",no1);
      for(j=0;j<k;j++)
      {
            printf("%d\n",arr[j]);
            if(count==1)
                  {mfact[j] = arr[j];mfactcnt++;}
            else
                  {nfact[j] = arr[j];nfactcnt++;}
      }
      count++;
}


void commonfactors()
{
      int i,j,min,ans=1;
      if(mfactcnt > nfactcnt)
      min = nfactcnt;
      else
      min = mfactcnt;

      for(i=0;i<mfactcnt;i++)
      {
            for(j=0;j<nfactcnt;j++)
            {
                  if(mfact[i] == nfact[j])
                  {
```

```
                        nfact[j] = 0;
                        ans = ans * mfact[i];
                        break;
                  }
                  else
                        continue;
            }
      }
      printf("ans:%d\n",ans);
}

int main(int argc,char **argv)
{
      int p,i,j,k=0;
      int x;
      int mf[MAX],nf[MAX],m,n;
      m = atoi(argv[1]);
      n = atoi(argv[2]);

      findprime(mf,m);
      findprime(nf,n);

      commonfactors();
      return 0;
}
```

Input Output:

```
[user1@centos LAB1]$ ./gcdPrime2.out 5 5

5's factor are:

5

5's factor are:

5

ans:5

[user1@centos LAB1]$ ./gcdPrime2.out 5 15

5's factor are:

5

15's factor are:

3

5

ans:5
```

```
[user1@centos LAB1]$ ./gcdPrime2.out 5 1

5's factor are:

5

1's factor are:

ans:1

[user1@centos LAB1]$
```

Complexity:

LAB2.

1.AIM. To implement Binary Search algorithm.

```
class BinarySearch
{
      static int arr[] = {1,2,3,4,5,6};
      public static void main(String args[])
      {

            int index = binarySearch(Integer.parseInt(args[0]));
            System.out.println("index of"+Integer.parseInt(args[0])+"="+index);

      }
      static int binarySearch(int key)
      {
            int l=0;
            int n = arr.length;
            int r=n-1;
            int mid;
            while(l<=r)
            {
                  mid = (l+r)/2;
                  if(arr[mid] == key)
                        return key;
                  else if(arr[mid] < key)
                        l = mid + 1;
                  else
                        r = mid -1;
            }
            return -1;
      }
}
```

Input output:

```
C:\Documents and Settings\jai shree krishna\Desktop\DAA_submission\LAB2>java Bin
arySearch 5

index of 5 = 5
```

Complexity: O(log n) where n is the number of elements in the array.

2.AIM. Insertion Sort

```
class InsertionSort
{
```

```
      public static int arr[]={1,5,4,3,2,6};
      public static void main(String args[])
      {
            InsrSort();
            for(int i=0;i<arr.length;i++)
            {
                  System.out.println("arr["+i+"]="+arr[i]);
            }
      }
      static void InsrSort()
      {
            int key = arr[0];
            int i,p;
            for(p=1;p<arr.length;p++)
            {
                  i = p;
                  key = arr[i];
                  while(key < arr[i-1] && i > 1)
                  {
                        arr[i] = arr[i-1];
                        i=i-1;
                  }
                  arr[i] = key;
            }
      }
}
```

Input output:

```
C:\Documents and Settings\jai shree krishna\Desktop\DAA_submission\LAB2>java Ins
ertionSort
arr[0]=1
arr[1]=2
arr[2]=3
arr[3]=4
arr[4]=5
arr[5]=6
```

Complexity:

3.AIM. To implement Matrix Multiplication

```
class MatrixMul
{
      static int a[][] = {{1,2},{3,4}};
      static int b[][] = {{4,3},{2,1}};
      static int c[][] = {{0,0},{0,0}};
      public static void main(String args[])
      {
            mulMatrix();
            for(int j = 0; j<2;j++)
```

```
            {
                  for(int k=0;k<2;k++)
                  {
                        System.out.println(c[j][k]);
                  }
            }

      }
      static void mulMatrix()
      {
            for(int i=0;i<2;i++)
            {
                  for(int j = 0; j<2;j++)
                  {
                        for(int k=0;k<2;k++)
                        {
                              c[i][j] = c[i][j] + (a[i][k] * b[k][j]);
                        }
                  }
            }
      }
}
```

Input output:

```
C:\Documents and Settings\jai shree krishna\Desktop\DAA_submission\LAB2>java Mat
rixMul
8
5
20
13
```

Complexity: O (n * n * n) where n is the size of matrix.

4.AIM. To find the Max Element from given array.

```
class MaxEle
{
      static int arr[] = {1,2,3,4,5,6};
      public static void main(String args[])
      {
            maxEle();
      }
      static void maxEle()
      {
            int max=arr[0];
            for(int i=0;i<arr.length;i++)
            {
                  if(arr[i] > max)
                  {
```

```
                        max = arr[i];
                    }
                }
                System.out.println("Max = "+max);
        }
}
```

Input output:

```
C:\Documents and Settings\jai shree krishna\Desktop\DAA_submission\LAB2>java Max
Ele
Max = 6
```

Complexity: O (n) where n is the number of elements in the given array.

5.AIM. Fibonacci using recursive

```
class RecursiveFibo
{
      public static void main(String args[])
      {
            int ele;
            for(int i = 0; i < Integer.parseInt(args[0]); i++)
            {
                  ele = fibo(i);
                  System.out.println(ele);
            }

      }
      static int fibo(int n)
      {
            int ans;
            if(n==0 || n ==1)
            {
                  return n;
            }
            else
            {
                  return (ans = fibo(n-1) + fibo(n-2));
            }
      }
}
```

Input output:

```
C:\Documents and Settings\jai shree krishna\Desktop\DAA_submission\LAB2>java Rec
ursiveFibo 5
0
1
1
2
```

3

Complexity:

6.AIM. Locker puzzle

```
#include<stdio.h>
#include<unistd.h>

#define MAX 2000



void initial(char *locker,int no)
{
      int i,j;
      for(i=0;i<MAX;i++)
      {
            locker[i]='c';
      }
      /*for(i=1;i<=no;i++)
      {
            printf("%c\t",locker[i]);
      }
      printf("\n");*/
}

void simulate(int no,char* locker)
{
      int i,j,k,l;
      printf("locker #\t\n");
      for(i=1;i<=no;i++)
      {
            printf("\t%d",i);
      }

printf("\n================================================================================
================\n");
      for(i=1;i<=no;i++)//no. of stds loop
      {
            printf("%dth std   ",i);
            for(j=1;j<=no;j++)//no. of lockers loop
            {
                  if(i==1)//std #1 opens every locker
                  {
                        for(k=1;k<=no;k++){
                              locker[k] = 'o';
                              printf("%c\t",locker[k]);
                        }
                        printf("\n");
                        break;
```

```
                }
                if(j%i == 0)//toggle if multiple of i
                {
                      if(locker[j] == 'o')
                      {
                            locker[j] = 'c';
                      }
                      else
                            locker[j] = 'o';
                }
                printf("%c\t",locker[j]);
          }
          printf("\n");
     }
}

int main(int argc,char** argv)
{
     int noOfStd;
     char locker[MAX];
     initial(locker,noOfStd);
     noOfStd = atoi(argv[1]);
     simulate(noOfStd,locker);
     return 0;
}
```

Input Output:

```
[user1@centos LAB2]$ ./lockerPuzz.out 10

locker #

        1       2       3       4       5       6       7       8       9       10
================================================================================
=======
1th std   o     o       o       o       o       o       o       o       o       o


2th std   o     c       o       c       o       c       o       c       o       c

3th std   o     c       c       c       o       o       o       c       c       c

4th std   o     c       c       o       o       o       o       o       c       c

5th std   o     c       c       o       c       o       o       o       c       o

6th std   o     c       c       o       c       c       o       o       c       o

7th std   o     c       c       o       c       c       c       o       c       o

8th std   o     c       c       o       c       c       c       c       c       o
```

```
9th std   o     c       c       o       c       c       c       c       o       o

10th std   o    c       c       o       c       c       c       c       o       c

[user1@centos LAB2]$
```

Complexity: O (n * n) where n is the number of students / locker.

LAB 3.

1.AIM. To implement Binary Adder.

```
class BinaryAdd
{
      static int A[] = {1,0,1,0};
      static int B[] = {1,1,1,1};
      public static void main(String args[])
      {
            int iCarry=0;
            int actCarry=0;
            int flag = 0;
            for(int i=A.length-1;i>=0;i--)
            {
                  if(iCarry == 0 && flag == 1)
                        A[i] = A[i] + actCarry;
                  iCarry = A[i] + B[i];
                  if(iCarry == 2)
                  {
                        flag = 1;
                        iCarry = 0;
                        actCarry = 1;
                  }
                  if(iCarry == 3)
                  {
                        flag = 1;
                        iCarry = 1;
                        actCarry = 1;
                  }
                  A[i] = iCarry;
            }
            for(int i=0;i<A.length; i++)
            {
                  System.out.println(A[i]);
            }
            System.out.println(actCarry);
      }
}
```

Input output:

```
C:\Documents and Settings\jai shree krishna\Desktop\DAA_submission\LAB3>java Bin
aryAdd
1
0
0
1
1
```

Complexity: O (n) where n is the number of bits to add.

2.AIM. Distance Array Problem

```
class DistanceArr
{
      static int A[] = {6,8,2,3,1};
      public static void main(String args[])
      {
            int min=999;
            int intans=0;
            int i=0;
            int flag = 0;
            while(i<A.length)
            {
                  for(int j=i+1;j<A.length;j++)
                  {
                        if(A[i] > A[j])//for avoiding -ve
                        {
                              intans = A[i]-A[j];
                              //System.out.println(intans+" " +A[i]+" "+A[j]);
                        }
                        else
                        {
                              intans = A[j]-A[i];
                              //System.out.println("here i m");
                              //System.out.println(intans+" " +A[j]+" "+A[i]);
                        }
                        if(intans <= min)
                        {
                              min = intans;
                              //System.out.println(min+"between"+A[j]+A[i]);
                        }
                  }
                  i++;
            }
            i=0;
            while(i<A.length)
            {
                  for(int j=i+1;j<A.length;j++)
                  {
                        if(A[i] > A[j])//for avoiding -ve
                        {
                              intans = A[i]-A[j];
                              //System.out.println(intans+" " +A[i]+" "+A[j]);
                        }
                        else
                        {
                              intans = A[j]-A[i];
                              //System.out.println("here i m");
                              //System.out.println(intans+" " +A[j]+" "+A[i]);
```

```
                        }
                        if(intans == min)
                        {
                              min = intans;
                              System.out.println(min+"between"+A[j]+A[i]);
                        }
                  }
                  i++;
            }

      }
}
```

Input output:

```
C:\Documents and Settings\jai shree krishna\Desktop\DAA_submission\LAB3>java Dis
tanceArr
1between32
1between12
```

Complexity: O (n * n) where n is the number of elements in array.

3.AIM. Inversion

```
class Inversion
{
      static int A[] = {2,3,8,6,1};
      public static void main(String args[])
      {
            for(int i=0;i<A.length;i++)
            {
                  for(int j=0;j<A.length;j++)
                  {
                        if(i<j && A[i] > A[j])
                        {
                              System.out.println("("+(i+1)+","+(j+1)+")");
                        }
                  }
            }
      }
}
```

Input output:

```
C:\Documents and Settings\jai shree krishna\Desktop\DAA_submission\LAB3>java Inv
ersion
(1,5)
(2,5)
(3,4)
(3,5)
(4,5)
```

Complexity: O (n * n) where n is the number of elements in array.

LAB 4.

1.AIM. To implement a Marge Sort algorithm.

```
class DArray
   {
   private long[] theArray;          // ref to array theArray
   private int nElems;               // number of data items

   public DArray(int max)            // constructor
      {
      theArray = new long[max];      // create array
      nElems = 0;
      }

   public void insert(long value)    // put element into array
      {
      theArray[nElems] = value;      // insert it
      nElems++;                      // increment size
      }

   public void display()             // displays array contents
      {
      for(int j=0; j<nElems; j++)    // for each element,
         System.out.print(theArray[j] + " ");  // display it
      System.out.println("");
      }

   public void mergeSort()           // called by main()
      {                              // provides workspace
      long[] workSpace = new long[nElems];
      recMergeSort(workSpace, 0, nElems-1);
      }

   private void recMergeSort(long[] workSpace, int lowerBound,
                                               int upperBound)
      {
      if(lowerBound == upperBound)            // if range is 1,
         return;                              // no use sorting
      else
         {                                    // find midpoint
         int mid = (lowerBound+upperBound) / 2;
                                              // sort low half
         recMergeSort(workSpace, lowerBound, mid);
                                              // sort high half
         recMergeSort(workSpace, mid+1, upperBound);
                                              // merge them
         merge(workSpace, lowerBound, mid+1, upperBound);
         }  // end else
```

```
      }  // end recMergeSort()

   private void merge(long[] workSpace, int lowPtr,
                          int highPtr, int upperBound)
      {
      int j = 0;                              // workspace index
      int lowerBound = lowPtr;
      int mid = highPtr-1;
      int n = upperBound-lowerBound+1;        // # of items

      while(lowPtr <= mid && highPtr <= upperBound)
         if( theArray[lowPtr] < theArray[highPtr] )
            workSpace[j++] = theArray[lowPtr++];
         else
            workSpace[j++] = theArray[highPtr++];

      while(lowPtr <= mid)
         workSpace[j++] = theArray[lowPtr++];

      while(highPtr <= upperBound)
         workSpace[j++] = theArray[highPtr++];

      for(j=0; j<n; j++)
         theArray[lowerBound+j] = workSpace[j];
      }  // end merge()

   }  // end class DArray

class Main
   {
   public static void main(String[] args)
      {
      int maxSize = 100;             // array size
      DArray arr;                    // reference to array
      arr = new DArray(maxSize);     // create the array

      arr.insert(64);                // insert items
      arr.insert(21);
      arr.insert(33);
      arr.insert(70);
      arr.insert(12);
      arr.insert(85);
      arr.insert(44);
      arr.insert(3);
      arr.insert(99);
      arr.insert(0);
      arr.insert(108);
      arr.insert(36);

      arr.display();                 // display items

      arr.mergeSort();               // merge sort the array
```

```
      arr.display();                  // display items again
      }  // end main()
   }  // end class MergeSortApp
```

Input output:

```
C:\Documents and Settings\jai shree krishna\Desktop\DAA_submission\LAB4>java Mai
n
64 21 33 70 12 85 44 3 99 0 108 36
0 3 12 21 33 36 44 64 70 85 99 108
```

Complexity: Θ (n log n) where n is the number of elements in the array.

2.AIM. To implement a Quick Sort algorithm

```
public class Main {

    static int partition(int arr[], int left, int right) {

      int i = left, j = right;

      int tmp;

      int pivot = arr[(left + right) / 2];


      while (i <= j) {

            while (arr[i] < pivot)

                  i++;

            while (arr[j] > pivot)

                  j--;

            if (i <= j) {

                  tmp = arr[i];

                  arr[i] = arr[j];

                  arr[j] = tmp;

                  i++;

                  j--;

            }
```

```
      }


      return i;

}



static void quickSort(int arr[], int left, int right) {

      int index = partition(arr, left, right);

      if (left < index - 1)

            quickSort(arr, left, index - 1);

      if (index < right)
            quickSort(arr, index, right);

}
    public static void main(String[] args) {
        int A[] = {10, 15, 26, 12, 9, 8, 22};
          for(int j=0; j<A.length; j++)    // for each element,
         System.out.print(A[j] + " ");  // display it
        System.out.println("");
        quickSort(A,0,A.length-1);
         for(int j=0; j<A.length; j++)    // for each element,
         System.out.print(A[j] + " ");  // display it
        System.out.println("");

    }
}
```

Input output:

```
C:\Documents and Settings\jai shree krishna\Desktop\DAA_submission\LAB4\quick ja
va>java Main
10 15 26 12 9 8 22
8 9 10 12 15 22 26
```

Complexity: Θ (n) where n, is n = n – p + 1 the number of elements in the array.

LAB 5.

1.AIM. Crossing the Bridge

```
class CrossBridge
{
      static int findminindex(int timming[])
      {
            int minindex = timming[0];
            for(int i=1;i<timming.length-1;i++)
            {
                  if(timming[i] < timming[minindex])
                  {
                        minindex = i;
                  }
            }
            System.out.println("min index:"+minindex);
            return minindex;
      }

      public static void main(String args[])
      {
            int count=5;
            int timming[]={2,1,5,10,12};
            int maxtime=0;
            int i=3;
            int minindex = findminindex(timming);
            while(count!=(timming.length-2))
            {
                  if(i-1 == minindex)
                  {
                        i--;
                        continue;
                  }
                  maxtime = maxtime + timming[i-1] + timming[minindex];
                  count--;
                  System.out.println(i-1+":is on the other bank");
                  i--;
            }
            maxtime = maxtime + timming[timming.length-1];
            System.out.println((timming.length-1)+":is on the other bank");
            System.out.println("time taken is:"+(maxtime-1));
      }
}
```

Input output:

```
C:\Documents and Settings\jai shree krishna\Desktop\DAA_submission\LAB5>java Cro
ssBridge
min index:1
2:is on the other bank
```

```
0:is on the other bank
4:is on the other bank
time taken is:20
```

Complexity:

2.AIM. Making a Change

```
class MakingChange
{
      public static void main(String args[])
      {
            int D[]={100,25,10,5,1};
            int amount=0;
            int sum = 14;
            int coin[]={0,0,0,0,0};
            int i=0;
            int flag = 1;
            while(i <= 4){
                  while(amount + D[i] <= sum)
                  {
                        amount = amount + D[i];
                        coin[i]++;
                        if(amount == sum)
                        {
                              for(int j=0;j<coin.length;j++)
                                    System.out.println(D[j]+"*"+coin[j]+"=     "+
(D[j]*coin[j]));
                              System.out.println("total  :"+sum);

                              flag = 0;
                              break;
                        }
                  }
                  if(flag == 0)
                        break;
                  i++;
            }
      }
}
```

Input output:

```
C:\Documents and Settings\jai shree krishna\Desktop\DAA_submission\LAB5>java Mak
ingChange
100*0=     0
25*0=     0
10*1=     10
5*0=     0
```

```
1*4=      4
total   :14
```

Complexity: O (n) where n is the number of dominations.

LAB 6.

1.AIM. To implement Kanpsack problem (fractional)

```
class Knapsack
{

      public static void main(String args[])
      {
            float weight = 0.0f;
            float wArr[] = {10.0f,20.0f,30.0f,40.0f,50.0f};
            float resultArr[] = {0.0f,0.0f,0.0f,0.0f,0.0f};
            float proArr[] = {20.0f,30.0f,66.0f,40.0f,60.0f};
            float ratioArr[] = {0.0f,0.0f,0.0f,0.0f,0.0f};
            float totalWeight = 100.0f;
            float totalProfit = 0.0f;
            for(int i=0;i<ratioArr.length;i++)
            {
                  ratioArr[i] = (float) ((float)proArr[i] / (float)wArr[i]);
            }
            for(int i=0;i<ratioArr.length;i++)
            {
                  for(int j=i+1;j<ratioArr.length;j++)
                  {
                        if(ratioArr[i] < ratioArr[j])
                        {
                              float temp = ratioArr[i];
                              ratioArr[i] = ratioArr[j];
                              ratioArr[j] = temp;

                              temp = wArr[i];
                              wArr[i] = wArr[j];
                              wArr[j] = temp;

                              temp = proArr[i];
                              proArr[i] = proArr[j];
                              proArr[j] = temp;
                        }
                  }
            }
            for(int i=0;i<ratioArr.length;i++)
            {
                  System.out.println(ratioArr[i]+"\t:\t"+proArr[i]
+"\t:\t"+wArr[i]);
            }

            for(int i=0;i<wArr.length;i++)
            {
                  if(weight+wArr[i] <= totalWeight)
                  {
```

```
                        totalProfit = totalProfit + proArr[i];
                        weight = weight + wArr[i];
                        resultArr[i]++;
                  }
                  else
                  {
                        totalProfit = totalProfit + (((totalWeight -
weight)/wArr[i])*proArr[i]);
                        resultArr[i] = ((totalWeight - weight)/wArr[i]);
                        break;
                  }
            }
            for(int i=0;i<resultArr.length;i++)
            {
                  System.out.println(resultArr[i]);
            }
            System.out.println("profilt="+totalProfit);
      }
}
```

Input output:

```
C:\Documents and Settings\jai shree krishna\Desktop\DAA_submission\LAB6>java Kna
psack
2.2     :       66.0    :       30.0
2.0     :       20.0    :       10.0
1.5     :       30.0    :       20.0
1.2     :       60.0    :       50.0
1.0     :       40.0    :       40.0
1.0
1.0
1.0
0.8
0.0
profilt=164.0
```

Complexity: O (n) + shorting time. Where n is the number of items.

2.AIM. Kruskal's Algorithm

```
import java.io.*;
import java.util.*;

public class Kruskal {
  private final int MAX_NODES = 21;
  private HashSet nodes[];               // Array of connected components
  private TreeSet allEdges;              // Priority queue of Edge objects
  private Vector allNewEdges;            // Edges in Minimal-Spanning Tree

  public static void main(String args[]) {
```

```
    System.out.println("[Kruskal] - 2011");
    if (args.length != 1) {
      System.out.println("Usage: java Kruskal <fileName>");
      System.exit(0);
    }
    Kruskal k = new Kruskal();
    k.readInGraphData(args[0]);
    k.performKruskal();
    k.printFinalEdges();
  }

  Kruskal() {
    // Constructor
    nodes = new HashSet[MAX_NODES];      // Create array for components
    allEdges = new TreeSet(new Edge());  // Create empty priority queue
    allNewEdges = new Vector(MAX_NODES); // Create vector for MST edges
  }

  private void readInGraphData(String fileName) {
    try {
      FileReader file = new FileReader(fileName);
      BufferedReader buff = new BufferedReader(file);
      String line = buff.readLine();
      while (line != null) {
        StringTokenizer tok = new StringTokenizer(line, " ");
        int from = Integer.valueOf(tok.nextToken()).intValue();
        int to   = Integer.valueOf(tok.nextToken()).intValue();
        int cost = Integer.valueOf(tok.nextToken()).intValue();

        allEdges.add(new Edge(from, to, cost));  // Update priority queue
        if (nodes[from] == null) {
          // Create set of connect components [singleton] for this node
          nodes[from] = new HashSet(2*MAX_NODES);
          nodes[from].add(new Integer(from));
        }

        if (nodes[to] == null) {
          // Create set of connect components [singleton] for this node
          nodes[to] = new HashSet(2*MAX_NODES);
          nodes[to].add(new Integer(to));
        }

        line = buff.readLine();
      }
      buff.close();
    } catch (IOException e) {
      //
    }
  }

  private void performKruskal() {
    int size = allEdges.size();
    for (int i=0; i<size; i++) {
      Edge curEdge = (Edge) allEdges.first();
```

```
      if (allEdges.remove(curEdge)) {
        // successful removal from priority queue: allEdges

        if (nodesAreInDifferentSets(curEdge.from, curEdge.to)) {
          // System.out.println("Nodes are in different sets ...");
          HashSet src, dst;
          int dstHashSetIndex;

          if (nodes[curEdge.from].size() > nodes[curEdge.to].size()) {
            // have to transfer all nodes including curEdge.to
            src = nodes[curEdge.to];
            dst = nodes[dstHashSetIndex = curEdge.from];
          } else {
            // have to transfer all nodes including curEdge.from
            src = nodes[curEdge.from];
            dst = nodes[dstHashSetIndex = curEdge.to];
          }

          Object srcArray[] = src.toArray();
          int transferSize = srcArray.length;
          for (int j=0; j<transferSize; j++) {
            // move each node from set: src into set: dst
            // and update appropriate index in array: nodes
            if (src.remove(srcArray[j])) {
              dst.add(srcArray[j]);
              nodes[((Integer) srcArray[j]).intValue()] = nodes[dstHashSetIndex];
            } else {
              // This is a serious problem
              System.out.println("Something wrong: set union");
              System.exit(1);
            }
          }

          allNewEdges.add(curEdge);
          // add new edge to MST edge vector
        } else {
          // System.out.println("Nodes are in the same set ... nothing to do
here");
        }

      } else {
        // This is a serious problem
        System.out.println("TreeSet should have contained this element!!");
        System.exit(1);
      }
    }
  }

  private boolean nodesAreInDifferentSets(int a, int b) {
    // returns true if graph nodes (a,b) are in different
    // connected components, ie the set for 'a' is different
    // from that for 'b'
    return(!nodes[a].equals(nodes[b]));
  }
```

```
  private void printFinalEdges() {
    System.out.println("The minimal spanning tree generated by "+
      "\nKruskal's algorithm is: ");
    while (!allNewEdges.isEmpty()) {
      // for each edge in Vector of MST edges
      Edge e = (Edge) allNewEdges.firstElement();
      System.out.println("Nodes: (" + e.from + ", " + e.to +
        ") with cost: " + e.cost);
      allNewEdges.remove(e);
    }
  }

  class Edge implements Comparator {
    // Inner class for representing edge+end-points
    public int from, to, cost;
    public Edge() {
      // Default constructor for TreeSet creation
    }
    public Edge(int f, int t, int c) {
      // Inner class constructor
      from = f; to = t; cost = c;
    }
    public int compare(Object o1, Object o2) {
      // Used for comparisions during add/remove operations
      int cost1 = ((Edge) o1).cost;
      int cost2 = ((Edge) o2).cost;
      int from1 = ((Edge) o1).from;
      int from2 = ((Edge) o2).from;
      int to1   = ((Edge) o1).to;
      int to2   = ((Edge) o2).to;

      if (cost1<cost2)
        return(-1);
      else if (cost1==cost2 && from1==from2 && to1==to2)
        return(0);
      else if (cost1==cost2)
        return(-1);
      else if (cost1>cost2)
        return(1);
      else
        return(0);
    }
    public boolean equals(Object obj) {
      // Used for comparisions during add/remove operations
      Edge e = (Edge) obj;
      return (cost==e.cost && from==e.from && to==e.to);
    }
  }

}
```

Input output:

```
C:\Documents and Settings\jai shree krishna\Desktop\DAA_submission\LAB6>java Kru
skal input.txt
[Kruskal] - 2011
The minimal spanning tree generated by
Kruskal's algorithm is:
Nodes: (1, 2) with cost: 1
Nodes: (2, 3) with cost: 2
Nodes: (6, 7) with cost: 3
Nodes: (4, 5) with cost: 3
Nodes: (4, 7) with cost: 4
Nodes: (2, 5) with cost: 4
```

Complexity: Θ ((n – 1) * 2 e f(2e,n)) where n is number of nodes, e is number of edges and f(2e,n) is the slowly growing function. And (e log n) > f(2e,n) so complexity can be considered as Θ (e log n).

LAB 7.

1.AIM. To implement Binomial theorem.

```
class Binomial
{
      public static void main(String args[])
      {
//          int ans=1;
            int f[][] = new int[Integer.parseInt(args[0])+2]
[Integer.parseInt(args[1])+2];//{1,0,0,0,0,0,0,0,0,0,1};
            if(Integer.parseInt(args[0]) < Integer.parseInt(args[1]) )
            {
                  System.out.println("0");
                  //exit(1);
            }
            for(int n=0;n<=Integer.parseInt(args[0])+1;n++)
            {
                  for(int k=0;k<=Integer.parseInt(args[1])+1;k++)
                  {
                        if(n==k)
                              f[n][k] = 1;
                  }
            }
            for(int n=0;n<=Integer.parseInt(args[0])+1;n++)
            {
                  for(int k=0;k<=Integer.parseInt(args[1])+1;k++)
                  {
                        if(n>k)
                              f[n][k] = 0;
                  }
            }

            for(int n=1;n<=Integer.parseInt(args[0])+1;n++)
            {
                  for(int k=1;k<=Integer.parseInt(args[1])+1;k++)
                  {
                        f[n][k] = f[n-1][k-1] + f[n-1][k];
                        //System.out.print(f[n][k]);
                  }
                  //System.out.println();
            }
            System.out.println( f[Integer.parseInt(args[0])+1]
[Integer.parseInt(args[1])+1] );

      }
}
```

Input output:

```
C:\Documents and Settings\jai shree krishna\Desktop\DAA_submission\LAB7>java Bin
```

```
omial 1 2
0
0

C:\Documents and Settings\jai shree krishna\Desktop\DAA_submission\LAB7>java Bin
omial 5 4
5

C:\Documents and Settings\jai shree krishna\Desktop\DAA_submission\LAB7>java Bin
omial 5 5
1
```

Complexity:

---

2. AIM. Fibonacci series

```
class Fibonacii
{
      public static void main(String args[])
      {
            //int i = Integer.parseInt(args[0]);
            int ans=1;
            int f[] = {0,1,0,0,0,0,0,0,0,0,0};
            for(int i=2;i<=Integer.parseInt(args[0])+1;i++)
            {
                  f[i] = f[i-1] + f[i-2];
            }
            System.out.println( f[Integer.parseInt(args[0])+1] );
      }
}
```

Input output:

```
 C:\Documents and Settings\jai shree krishna\Desktop\DAA_submission\LAB7>java Fib
onacii 5
8
```

Complexity:

---

3. AIM. To write a program for finding Longest Common Sub-sequence.

```
class LCS
{
```

```
	public static void main(String args[])
	{
		char x[] = {'x','A','B','C','B','D','A','B'};
		char y[] = {'x','B','D','C','A','B','A'};
		int m=x.length;
		int n=y.length;
		int c[][] = new int[m][n];
		for(int i=0; i<n; i++)
		{
			c[0][i] = 0;
		}
		for(int j=0; j<m; j++)
		{
			c[j][0] = 0;
		}
		for(int i=1;i<m;i++)
		{
			for(int j=1;j<n;j++)
			{
				if(x[i] == y[j])
				{
					c[i][j] = c[i-1][j-1] + 1;
				}
				else
				{
					if(c[i-1][j] >= c[i][j-1])
						c[i][j] = c[i-1][j];
					else
						c[i][j] = c[i][j-1];
				}
			}
		}
		for(int i=0;i<m;i++)
		{
			for(int j=0;j<n;j++)
			{
				System.out.print(c[i][j]);
			}
			System.out.println();
		}
		System.out.println(c[m-1][n-1]);
	}
}
```

Input output:

```
C:\Documents and Settings\jai shree krishna\Desktop\DAA_submission\LAB7>java LCS

0000000
0000111
0111122
0112222
0112233
0122233
```

```
0122334
0122344
4
```

Complexity: O (m * n) where m is the length of string 1 and n is the length of string 2.

LAB 8.

1.AIM. To find All pair shortest path

```
class AllPairShortestPath
{
      static int D0[][] = { {0,5,999,999},
                  {50,0,15,5},
                  {30,999,0,15},
                  {15,999,5,0} };
      static int p[][] = new int[4][4];
      public static void main(String args[])
      {
            for(int i=0;i<4;i++)
            {
                  for(int j=0;j<4;j++)
                  {
                        p[i][j] = 0;
                  }
            }
            for(int k=0;k<4;k++)
            {
                  for(int i=0;i<4;i++)
                  {
                        for(int j=0;j<4;j++)
                        {
                              if(D0[i][j] > (D0[i][k] + D0[k][j]) )
                              {
                                    D0[i][j] = D0[i][k] + D0[k][j];
                                    p[i][j] = k+1;
                              }
                        }
                  }
            }
            for(int i=0;i<4;i++)
            {
                  for(int j=0;j<4;j++)
                  {
                        System.out.print(" "+D0[i][j]);
                  }
                  System.out.println();
            }
            System.out.println();
            for(int i=0;i<4;i++)
            {
                  for(int j=0;j<4;j++)
                  {
                        System.out.print(" "+p[i][j]);
                  }
                  System.out.println();
```

```
            }
      }
}
```

Input output:

```
C:\Documents and Settings\jai shree krishna\Desktop\DAA_submission\LAB8>java All
PairShortestPath
 0 5 15 10
 20 0 10 5
 30 35 0 15
 15 20 5 0

 0 0 4 2
 4 0 4 0
 0 1 0 0
 0 1 0 0
```

Complexity: O (n * n * n) where n is the number of cities.

2.AIM. Matrix Chain Multiplication

```
class MCM
{
      static int P[] = {30,35,15,5,10,20};
      static int s[][] = new int[P.length][P.length];
      public static void main(String args[])
      {
            int n = P.length-1;//6-1=5
            int j = 0;
            int m[][] = new int[P.length][P.length];
            int i=0,k=0,l=0,t=0,ans=0;
            for(i=0;i<n;i++)
            {
                  m[i][i] = 0;
            }

            for(l=1;l<=n-1;l++)
            {
                  //System.out.println("l="+l+"  ");
                  for(i=0;i<=(n-(l+2)+1);i++)
                  {
                        //System.out.print("i="+i+"   ");
                        j=(l+1)+i-1; m[i][j] = 999999;
                        //System.out.print("j="+j+"   ");
                        for(k=i;k<=j-1;k++)
                        {
                              t = m[i][k] + m[k+1][j] + (P[i]*P[k+1]*P[j+1]);
                              //System.out.print((P[i]*P[k+1]*P[j+1]));

                              //System.out.print("m["+i+"]"+"["+k+"]="+m[i][k]
```

```
+"+");
                                    //System.out.print("m["+(k+1)+"]"+"["+j+"]="+m[k+1]
[j]);
                                    //System.out.print("\t"+"m["+i+"]"+"["+j+"]="+t+"\t")
;
                                    //System.out.print("k="+k+"   ");
                                    ans = t;
                                    if(t < m[i][j])
                                    {
                                          m[i][j] = t;
                                          s[i][j] = k;
                                    }
                              }
                              //System.out.println();
                        }
                        //System.out.println();
                  }
                  System.out.println(" "+m[0][4]);
                  for(i=1;i<P.length;i++)
                  {
                        for(j=1;j<P.length;j++)
                        {
                              if(i<=j)
                              System.out.print(" "+s[i][j]);
                        }
                        System.out.println();
                  }
            }
}
```

Input output:

```
C:\Documents and Settings\jai shree krishna\Desktop\DAA_submission\LAB8>java MCM

 11875
 0 1 2 2 0
 0 2 2 0
 0 3 0
 0 0
 0
```

Complexity: O (n * n * n) where n is number of matrix to multiply.

LAB 9.

1.AIM. Memory Function Knapsack (0/1)

```
class MemFuncKnapsack
{
	public static int weight = 0;
	public static int wArr[] = {0,2,1,3,2};
	public static int resultArr[] = {0,0,0,0,0};
	public static int proArr[] = {0,12,10,20,15};
	public static int vArr[][] = new int[5][6];//{-1,-1,-1,-1,-1};
	public static int totalWeight = 5;
	public static int totalProfit = 0;

	public static void main(String args[])
	{
		for(int k=0;k<5;k++)
		{
			for(int l=0;l<6;l++)
			{
				vArr[k][l] = -1;
			}
		}
		vArr[0][0] = 0;
		vArr[1][0] = 0;
		vArr[2][0] = 0;
		vArr[3][0] = 0;
		vArr[4][0] = 0;
		vArr[0][1] = 0;
		vArr[0][2] = 0;
		vArr[0][3] = 0;
		vArr[0][4] = 0;
		vArr[0][5] = 0;


		System.out.println(MF_Knapsack(4,5));
		for(int k=0;k<5;k++)
		{
			for(int l=0;l<6;l++)
			{
				System.out.print("\t"+vArr[k][l]+" ");
			}
			System.out.println();
		}
	}
	public static int max(int i,int j)
	{
		if(i>=j)
		{
			return i;
		}
```

```
            return j;
      }
      public static int MF_Knapsack(int i,int j)
      {
            if(vArr[i][j] < 0)
            {
                  if(j < wArr[i])
                  {
                        MF_Knapsack(i-1,j);
                  }
                  else
                  {
                        System.out.println("i="+i+"j="+j);
                        int val = max(MF_Knapsack(i-1,j),(proArr[i]+MF_Knapsack(i-
1,j-wArr[i]) ));
                        vArr[i][j] = val;
                  }
            }

            return vArr[i][j];
      }
}
```

Input output:

```
C:\Documents and Settings\jai shree krishna\Desktop\DAA_submission\LAB9>java Mem
FuncKnapsack
i=4j=5
i=3j=5
i=2j=5
i=1j=5
i=1j=4
i=2j=2
i=1j=2
i=3j=3
i=2j=3
i=1j=3
37
        0       0       0       0       0       0
        0       -1      12      12      12      12
        0       -1      12      22      -1      22
        0       -1      -1      22      -1      32
        0       -1      -1      -1      -1      37
```

Complexity:

2.AIM. Sum Of Subsets

```
class Sos{
```

```
	public static int w[] = {7,11,13,24};
	public static int x[] = {0,0,0,0};
	public static int m = 31;
	public static void main(String args[])
	{
		sos(0,0,55);
	}
	public static void sos(int s,int k,int r)
	{
		x[k] = 1;
		if(s+w[k] == m)
		{
			for(int i=0;i<x.length;i++)
			{
				System.out.println(x[i]);
			}
		}
		else if(s+w[k]+w[k+1] <= m)
		{
			sos(s+w[k],k+1,r-w[k]);
		}
		if(s+r-w[k] >= m && s+w[k+1] <= m)
		{
			x[k] = 0;
			sos(s,k+1,r-w[k]);
		}
	}
}
```

Input output:

```
C:\Documents and Settings\jai shree krishna\Desktop\DAA_submission\LAB9>java Sos

1
1
1
0

1
0
0
1
```

Complexity: O (p(n) * 2^n) or O (q(n) * n!) where n is the number of nodes in back tracking tree. And p(n) and q(n) are bounding functions.

LAB10:

1.AIM: To implement Fifteen Puzzle problem

```
import java.util.HashMap;
import java.util.LinkedList;
import java.util.Map;
import java.util.Queue;

class FifteenPuzzle {

    Map<String,String> stateHistory = new HashMap<String,String>(); // relates
each position to its predecessor
    Map<String,Integer> stateDepth = new HashMap<String,Integer>();
    Queue<Integer[][]> agenda=new LinkedList<Integer[][]>();
    final int GRIDSIZE=4;
    int row=0,col=0;
    public static void main(String args[]){

        Integer init[][]={{1,2,3,4},{5,6,0,8},{9,10,7,11},{13,14,15,12}};

        FifteenPuzzle e = new FifteenPuzzle();              // New Instance of the
EightPuzzle

        e.add(init,null);                                                    // Add
the Initial State

        while(!e.agenda.isEmpty()){
            Integer[][] currentState = e.agenda.remove();
            e.up(currentState);                                          // Move the
blank space up and add new state to queue
            e.down(currentState);                                        // Move the
blank space down
            e.left(currentState);                                        // Move left
            e.right(currentState);                              // Move right and
remove the current node from Queue
        }

        System.out.println("Solution doesn't exist");
    }

    //Add method to add the new Integer[][] to the Map and Queue
    void add(Integer newState[][], Integer oldState[][]){
        if(!stateDepth.containsKey(convertToString(newState))){
            int newValue = oldState == null ? 0 :
stateDepth.get(convertToString(oldState)) + 1;
            stateDepth.put(convertToString(newState), newValue);
            agenda.add(newState);
            stateHistory.put(convertToString(newState),
convertToString(oldState));
```

```
        }
    }

    /* Each of the Methods below Takes the Current State of Board as Integer[][].
Then the operation to move the blank space is done if possible.
      After that the new Integer[][] is added to the map and queue.If it is the
Goal State then the Program Terminates.
     */
    void up(Integer[][] currentState){
        Integer[][] nextState=new Integer[GRIDSIZE][GRIDSIZE];
        getIndicesOfZero(currentState, nextState);
        if(row!=0){
            nextState[row-1][col]=currentState[row][col];
            nextState[row][col]=currentState[row-1][col];
            checkCompletion(currentState, nextState);
        }
    }


    void down(Integer[][] currentState){
        Integer[][] nextState=new Integer[GRIDSIZE][GRIDSIZE];
        getIndicesOfZero(currentState, nextState);
        if(row!=GRIDSIZE-1){
            nextState[row+1][col]=currentState[row][col];
            nextState[row][col]=currentState[row+1][col];
            checkCompletion(currentState, nextState);
        }
    }
    void left(Integer[][] currentState){
        Integer[][] nextState=new Integer[GRIDSIZE][GRIDSIZE];
        getIndicesOfZero(currentState, nextState);
        if(col!=0){
            nextState[row][col-1]=currentState[row][col];
            nextState[row][col]=currentState[row][col-1];
            checkCompletion(currentState, nextState);
        }
    }
    void right(Integer[][] currentState){
        Integer[][] nextState=new Integer[GRIDSIZE][GRIDSIZE];
        getIndicesOfZero(currentState, nextState);
        if(col!=GRIDSIZE-1){
            nextState[row][col+1]=currentState[row][col];
            nextState[row][col]=currentState[row][col+1];
            checkCompletion(currentState, nextState);
        }
    }

    private void checkCompletion(Integer[][] oldState, Integer[][] newState) {
        add(newState, oldState);
        Integer[][] completeState={{1,2,3,4},{5,6,7,8},{9,10,11,12},{13,14,15,0}};
        boolean equality=true;
        outer:for(int i=0;i<GRIDSIZE;i++){
            for(int j=0;j<GRIDSIZE;j++){
                if(newState[i][j]!=completeState[i][j]){
```

```
                        equality=false;
                        break outer;
                    }
                }
            }

            if(equality){
                System.out.println("Solution Exists at Level
"+stateDepth.get(convertToString(newState))+" of the tree");
                String traceState = convertToString(newState);
                while (traceState != null) {
                    System.out.println(traceState + " at " +
stateDepth.get(traceState));
                    traceState = stateHistory.get(traceState);
                }
                System.exit(0);

            }
        }
        String convertToString(Integer[][] a){
            String str="";
            if(a!=null){
                    for(int i=0;i<GRIDSIZE;i++){
                            for(int j=0;j<GRIDSIZE;j++){
                                    str+=a[i][j];
                            }
                    }
            }
            else{
                    str=null;
            }
            return str;
        }
        void getIndicesOfZero(Integer[][] currentState,Integer[][] nextState){
            for(int i=0;i<GRIDSIZE;i++){
                for(int j=0;j<GRIDSIZE;j++){
                    nextState[i][j]=currentState[i][j];
                }
            }
            outer:for(int i=0;i<GRIDSIZE;i++){
                for(int j=0;j<GRIDSIZE;j++){
                    if(currentState[i][j]==0){
                            row=i;
                            col=j;
                            break outer;
                    }
                }
            }

        }
}
```

Input output:

```
C:\Documents and Settings\jai shree krishna\Desktop\DAA_submission\LAB10>java Fi
fteenPuzzle
Solution Exists at Level 3 of the tree
1234567891011121314150 at 3
1234567891011013141512 at 2
1234567891001113141512 at 1
1234560891071113141512 at 0
```

Complexity: O (p(n) * 2^n) or O (q(n) * n!) where n is the number of nodes in back tracking tree. And p(n) and q(n) are bounding functions.

2.AIM: To implement graph coloring problem

```
class GraphColoring
{
      public static int x[] = {0,0,0};
      /*public static int A[][] = {{0,1,1,0,1},
                        {1,0,1,0,1},
                        {1,1,0,1,0},
                        {0,0,1,0,1},
                        {1,1,0,1,0}};*/
      public static int A[][] = {{0,1,1},
                        {1,0,1},
                        {1,1,0},
                        };
      public static int m = 3;
      public static int n = 2;
      public static void main(String args[])
      {
            mColoring(0);
      }
      public static void mColoring(int k)
      {
            while(true)
            {

                  nextValue(k);
                        //System.out.println(x[k]);
                  if(x[k] == 0)
                  {
                        //System.out.println("no coloring is possible");
                        return;
                  }
                  if(k==n)
                  {
                        for(int i=0;i<x.length;i++)
                        {
                              System.out.print(x[i] + " ");
                        }
```

```
                        System.out.println();
                  }
                  else
                        mColoring(k+1);
            }

      }
      public static void nextValue(int k)
      {
            int j=0;
            while(true)
            {
                  x[k] = (x[k] + 1) % (m+1);
                  if(x[k] == 0) return;
                  for(j=0;j<n;j++)
                  {
                        if( (A[k][j] != 0) && x[k] == x[j])
                        {
                              break;
                        }
                  }
                  if(j==n){
                  return; }
            }
      }
}
```

Input output:

```
C:\Documents and Settings\jai shree krishna\Desktop>java GraphColoring
1 2 3
1 3 2
2 1 3
2 3 1
3 1 2
3 2 1
```

Complexity: size of tree = $\Sigma_{(i=0 \text{ to } n-1)}$ (m^i)
complexity of bounding function O (m * n) where m is the number of colors and n is the number of nodes in graph.
Therefor, complexity is: size of tree * complexity of bounding function.

LAB11:

1. AIM: To implement Horpool's String Matching Alg

```
class Horspool

{

      static int table[] = new int[26];

      //static char pattern[] = {'B','A','R','B','E','R'};

      //static char text[] =
{'J','I','M','S','A','W','M','E','I','N','A','B','A','R','B','E','R','S','H','O','
P'};

      static char pattern[] = {'A'};

      static char text[] =
{'J','I','M','S','A','W','M','E','I','N','A','B','A','R','B','E','R','S','H','O','
P'};

      public static void main(String args[])

      {

            int m = pattern.length;

            for(int j=0; j<table.length-1; j++)

            {

                  table[j] = pattern.length;

            }



            for(int j=0; j<=m-2; j++)

            {



                  table[((int)pattern[j])%65] = m-1-j;

                  System.out.println("for pattern char:"+pattern[j]
+table[((int)pattern[j])%65]);

            }
```

```
            int i = m-1;
            int n = text.length;
            while(i <= n-1)
            {
                  int k = 0;
                  while( k < m && pattern[m-k-1] == text[i-k])
                  {
                        k = k + 1;
                  }
                  if(k == m){
                        System.out.println("pattern occurs at:"+(i-m+2));
                        i = i + table[((int)text[i])%65];
                  }
                  else{
                        i = i + table[((int)text[i])%65];
                  }
            }
      }
}
```

Input Output:

```
[user1@centos LAB11]$ javac Horspool.java
[user1@centos LAB11]$ java Horspool
pattern occurs at:5
pattern occurs at:11
pattern occurs at:13
```

Complexity: Θ (m * n) where m is length of pattern and n is the length of text.