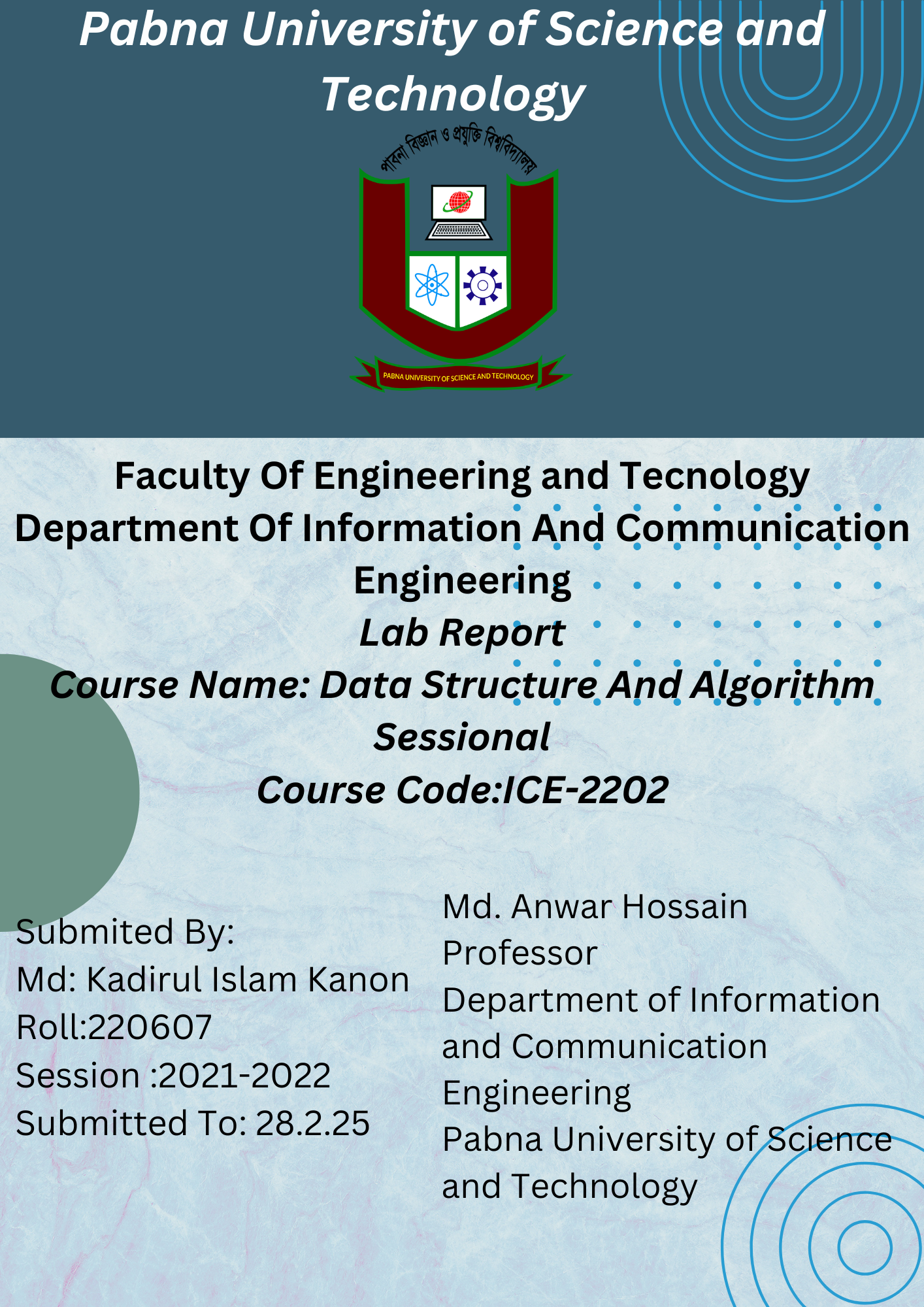
******

|  |  |
| --- | --- |
| **Sl.** | **Problem Statement** |
| **1.** | |  | | --- | | Write a program to sort a linear array using the bubble sort algorithm. | |
| **2.** | |  | | --- | | Write a program to find an element using a linear search algorithm. | |
| **3.** | |  | | --- | | Write a program to sort a linear array using the merge sort algorithm. | |
| **4.** | |  | | --- | | Write a program to find an element using the binary search algorithm. | |
| **5.** | |  | | --- | | Write a program to find a given pattern from text using the pattern matching algorithm. | |
| **6.** | |  | | --- | | Write a program to solve **n** queen's problem using backtracking. | |
| **7.** | |  | | --- | | Consider a set **S = {5,10,12,13,15,18}** and **d = 30**. Write a program to solve the sum of subset problem. | |
| **8.** | |  | | --- | | Write a program to solve the following **0/1 Knapsack** using dynamic programming approach **profits P = (15,25,13,23), weight W = (2,6,12,9), Knapsack C = 20**, and the number of items **n=4**. | |
| **9.** | |  | | --- | | Write a program to solve the **Tower of Hanoi** problem for the **N** disk. | |

**Index**

Problem 1:

#include<bits/stdc++.h>

using namespace std;

int main()

{

   int n;

   cin>>n;

   int arr[n];

   for(int i=0;i<n;i++)

   {

       cin>>arr[i];

   }

       for(int i=0;i<n-1;i++)

       {

          for(int j=0;j<n-1-i;j++)

           if(arr[j]>arr[j+1])

           {

               int tmp=arr[j];

               arr[j]=arr[j+1];

               arr[j+1]=tmp;

           }

       }

   for(int i=0;i<n;i++)

   {

       cout<<arr[i]<<" ";

   }

   cout<<endl;

}

**Output**:

5

1 8 3 0 2

0 1 2 3 8

Problem 2:

#include<bits/stdc++.h>

using namespace std;

int linearsearch(int arr[],int n, int key)

{

    for(int i=0;i<n;i++)

    {

        if(arr[i]==key)

        {

            return i;

        }

    }

    return -1;

}

int main()

{

   int n;

   cin>>n;

   int arr[n];

   for(int i=0;i<n;i++)

   {

       cin>>arr[i];

   }

   int key;

   cin>>key;

   cout<<linearsearch(arr,n,key)<<endl;

}

**Output**:

5

1 7 4 9 3

4

2

Problem 3:

#include<bits/stdc++.h>

using namespace std;

int binarysearch(int arr[],int n,int key)

{

    int s=0;

    int e=n;

    while(s<=e)

    {

        int mid=(s+e)/2;

        if(arr[mid]==key)

        {

            return mid;

        }

        else if(arr[mid]>key)

        {

            e=mid-1;

        }

        else

        {

            s=mid+1;

        }

    }

    return -1;

}

int main()

{

    int n;

   cin>>n;

   int arr[n];

   for(int i=0;i<n;i++)

   {

       cin>>arr[i];

   }

   int key;

   cin>>key;

   cout<<binarysearch(arr,n,key)<<endl;

}

**Output**:

5

1 2 3 4 5

4

3

Problem 4:

#include<bits/stdc++.h>

using namespace std;

int n,arr[1000005],temp[10005];

void merge(int l,int r){

    int mid=(l+r)/2;

    int i1=l,i2=mid+1,j=l;

    while(i1<=mid &&i2<=r)

    {

        if(arr[i1]<arr[i2])

        {

            temp[j]=arr[i1];

            i1++;

        }

        else{

            temp[j]=arr[i2];

            i2++;

        }

        j++;

    }

    while (i1<=mid){

        temp[j]=arr[i1];

        j++,i1++;

    }

    while (i2<=r){

        temp[j]=arr[i2];

        j++,i2++;

    }

    for(int j=l;j<=r;j++)

    {

       arr[j]=temp[j];

    }

}

void mergeSort(int l,int r)

{

    if(l==r)return;

    int mid=(l+r)/2;

    mergeSort(l,mid);

    mergeSort(mid+1,r);

    merge(l,r);

}

int main()

{

    cin>>n;

    for(int i=0;i<n;i++)

    {

        cin>>arr[i];

    }

    mergeSort(0,n-1);

    for(int i=0;i<n;i++)

    {

        cout<<arr[i]<<" ";

    }

}

**Output:**

5

1 9 3 7 2

1 2 3 7 9

Problem 5:

#include<bits/stdc++.h>

using namespace std;

void patternSearch(string text, string pattarn){

int n = text.size();

int m = pattarn.size();

for(int i= 0; i <= n-m ; i++){

int j = 0;

while(j<m && text[i+j] == pattarn[j]){

j++;

}

if(j ==  m){

cout << " pattern found at:" << i << endl;

}

}

}

int main(){

string text = "AABBCCCCCCCAABBCCCCAABB";

string pattarn = "AABB";

patternSearch(text,pattarn);

}

**Output:**

Pattern fount at:0

Pattern found at:11

Pattern found at:19

Problem 7:

#include <iostream>

#include <cmath>

using namespace std;

int main() {

    int N, target\_sum;

    cout << "Enter the number of elements: ";

    cin >> N;

    int S[N];

    cout << "Enter the elements: ";

    for (int i = 0; i < N; i++) {

        cin >> S[i];

    }

    cout << "Enter the target sum: ";

    cin >> target\_sum;

    int total\_subsets = 1 << N;

    int count = 0;

    for (int mask = 0; mask < total\_subsets; mask++) {

        int subset\_sum = 0;

        bool found = false;

        for (int j = 0; j < N; j++) {

            if (mask & (1 << j)) {

                subset\_sum += S[j];

            }

        }

        if (subset\_sum == target\_sum) {

            found = true;

            cout << "{ ";

            for (int j = 0; j < N; j++) {

                if (mask & (1 << j)) {

                    cout << S[j] << " ";

                }

            }

            cout << "}\n";

            count++;

        }

    }

    cout << "Total subsets found: " << count << endl;

    return 0;

}

**Output** :

Enter the number of elements: 6

Enter the elements: 5 10 12 13 15 18

Enter the target sum: 30

{5 12 13}

{5 10 15}

{12 18}

Total subsets found: 3

Problem 8:

#include <iostream>

using namespace std;

#define N 4

void printSolution(int placed[]) {

    static int solutionCount = 0;

    cout << "\nSolution " << ++solutionCount << ":\n";

    for (int i = 0; i < N; i++, cout << "\n")

        for (int j = 0; j < N; j++)

            cout << (placed[i] == j ? 'Q' : '.') << " ";

}

bool isSafe(int placed[], int row, int col) {

    for (int prev = 0; prev < row; prev++) {

        if (placed[prev] == col ||

            placed[prev] - prev == col - row ||

            placed[prev] + prev == col + row) {

            return false;

        }

    }

    return true;

}

void solveNQueens(int placed[], int row) {

    if (row == N) {

        printSolution(placed);

        return;

    }

    for (int col = 0; col < N; col++) {

        if (isSafe(placed, row, col)) {

            placed[row] = col;

            solveNQueens(placed, row + 1);

        }

    }

}

int main() {

    int placed[N] = {-1};

    solveNQueens(placed, 0);

    return 0;

}

**Output**:

Solution 1:

. Q . .

. . . Q

Q . . .

. . Q .

Solution 2:

. . Q .

Q . . .

. . . Q

. Q . .

Problem 9:

#include<iostream>

using namespace std;

void tower(int n,char s,char h,char d)

{

    if(n==0)return;

    tower (n-1,s,d,h);

    cout<<s<<"-"<<d<<endl;

    tower (n-1,h,s,d);

    return;

}

int main()

{

    tower(3,'A','C','D');

}

Output:

A-D

A-C

D-C

A-D

C-A

C-D

A-D