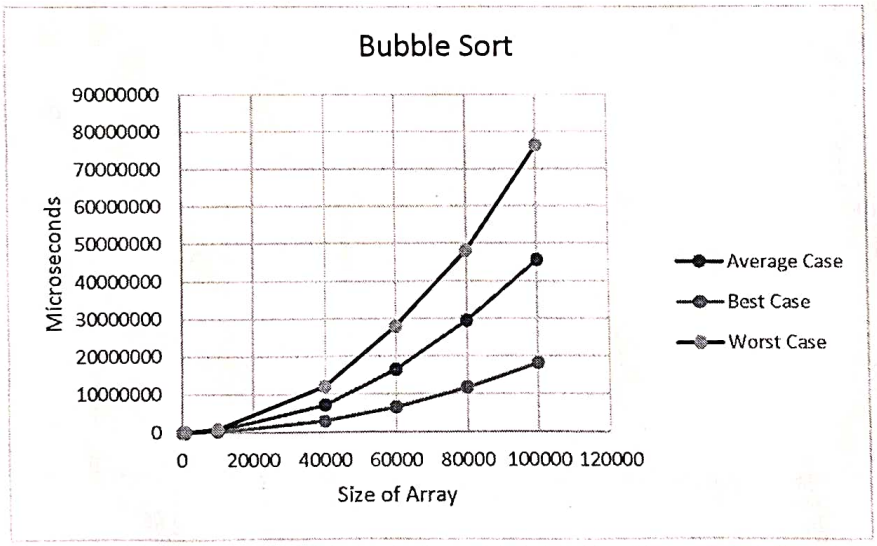

Bubble Sort
90000000
80000000
70000000
60000000
50000000
40000000
30000000
20000000
10000000
0
Microseconds
0
20000
40000
60000
80000
100000
120000
Size of Array
Average Case
Best Case
Worst Case

Date

Expt. No. .....

Page No. .....

Experiment-1(a)

Bubble, Selection and Insertion Sort.

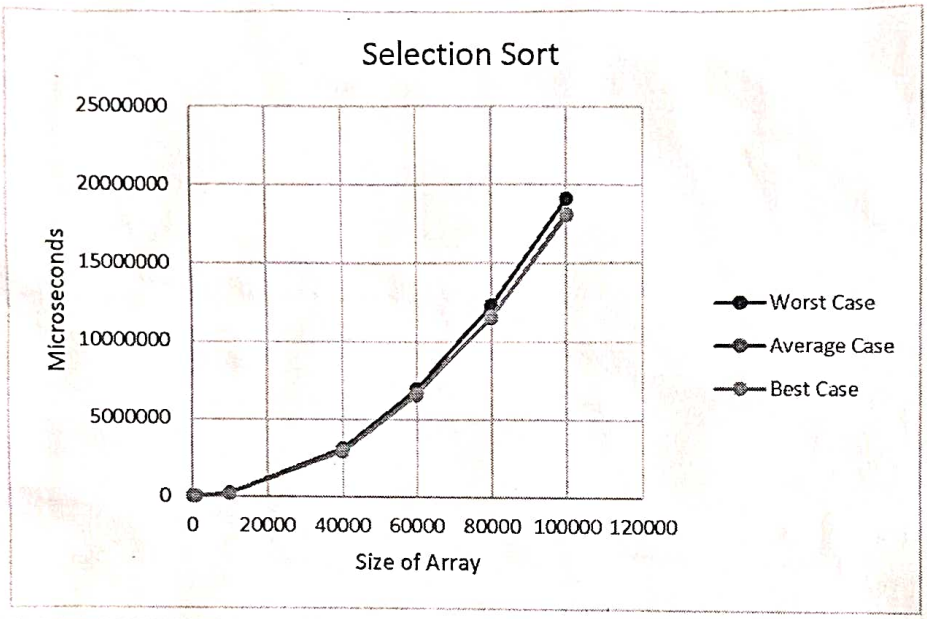
Program:-

```
#include <bits/stdc++.h>
#include <chrono>
using namespace std;
using namespace std::chrono;

void printarray(int a[], int n){
    fstream file;
    file.open("output.txt", ios::out);
    for(int i=0; i<n; i++){
        file<<a[i]<<' ';
    }
    file.close();
}

void bubbleSort(int a[], int n){
    int flag = 0;
    for(int i=0; i<n; i++){
        flag = 0;
        for(int j=0; j<n-1-i; j++){
            if(a[j]>a[j+1]){
```

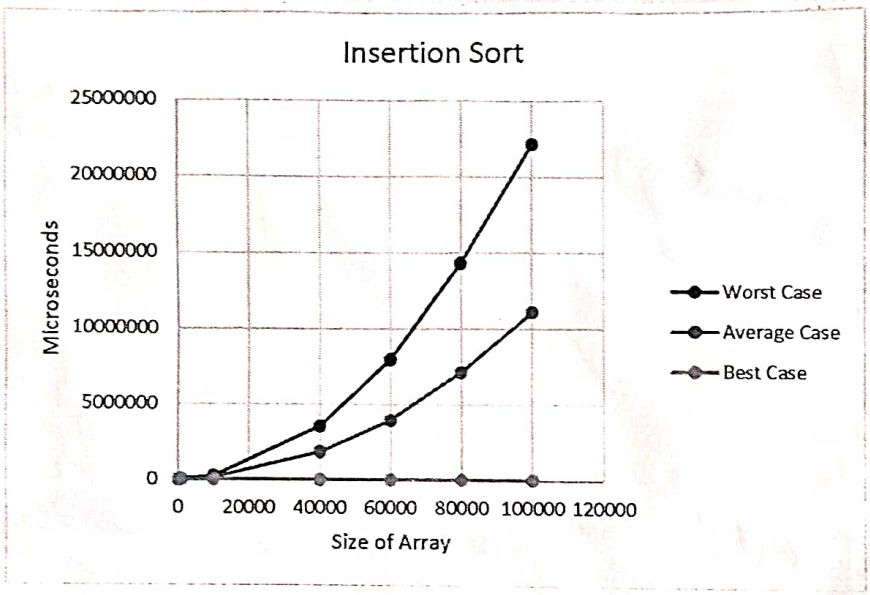
Teacher's Signature : _____


Selection Sort
25000000
20000000
15000000
10000000
5000000
0
Microseconds
0
20000
40000
60000
80000
100000
120000
Size of Array
Worst Case
Average Case
Best Case

```
            swap(a[j], a[j+1]);
            flag = 1;
        }
    }
    if(flag == 0) break;
}
printarray(a,n);
}

void selectionSort(int a[], int n){
    for(int i=0; i<n; i++){
        int mi = i;
        for(int j=i+1; j<n; j++){
            if(a[mi] > a[j])
                mi = j;
            }
        }
        swap(a[i], a[mi]);
    }
    printarray(a,n);
}

void insertionSort(int a[], int n){
    for(int i=1; i<n; i++){
        int k = a[i];
```


Insertion Sort
25000000
20000000
15000000
10000000
5000000
0
Microseconds
0
20000
40000
60000
80000
100000
120000
Size of Array
Worst Case
Average Case
Best Case

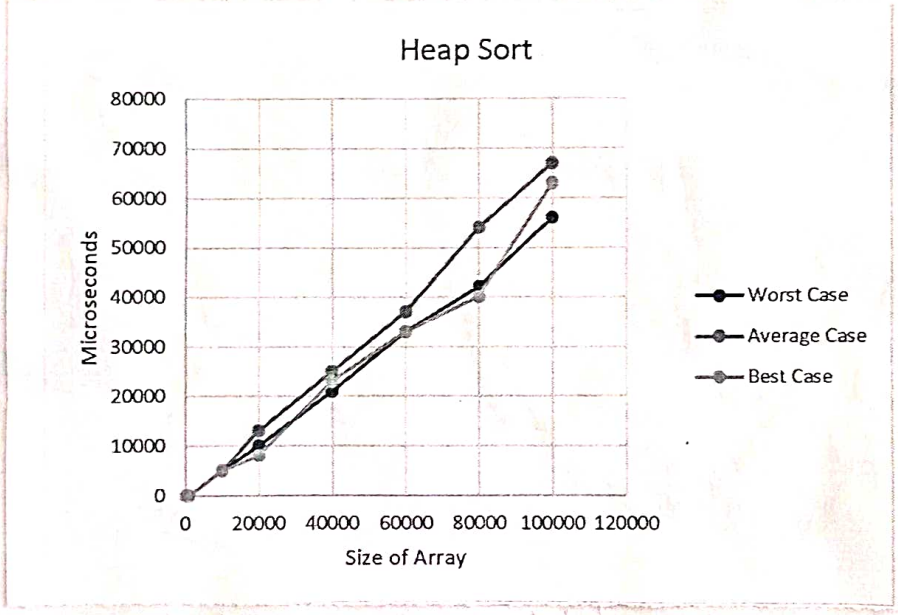
Date

Expt. No. Page No.

```
        int j = i-1;
        while(j>=0 && a[j] >K)
        {
            a[j+1] =a[j];
            j--;
        }
        a[j+1] = K;
    }
    printarray(a,n);
}

int main(){
    int n=100;
    int a[n+100];
    for(int i=0; i<n; i++) a[i] = i;
    auto start =high_resolution_clock::now();
    bubbleSort(a,n);
    auto stop = high_resolution_clock::now();
    auto duration = duration_cast<microseconds>(stop-start);
    cout << "Time taken by function: " << duration.count()
         << " microseconds" <<endl;
}
```

Teacher's Signature :


Heap Sort
80000
70000
60000
50000
40000
30000
20000
10000
0
Microseconds
0
20000
40000
60000
80000
100000
120000
Size of Array
Worst Case
Average Case
Best Case

Date .....

Expt. No. ..... Page No. .....

Experiment 1(b)

Heap Sort:-

Program:-

```
#include <bits/stdc++.h>
#include <chrono>
using namespace std;
using namespace std::chrono;

void heapify(int a[], int n, int i){
    int largest = i;
    int l = 2*i + 1;
    int r = 2*i + 2;
    if (l<n && a[l]>a[largest]){
        largest = l;
    }
    if (r<n && a[r] > a[largest]){
        largest = r;
    }
    if (largest != i){
        swap(a[i], a[largest]);
        heapify(a, n, largest);
    }
}
```

Teacher's Signature : _____

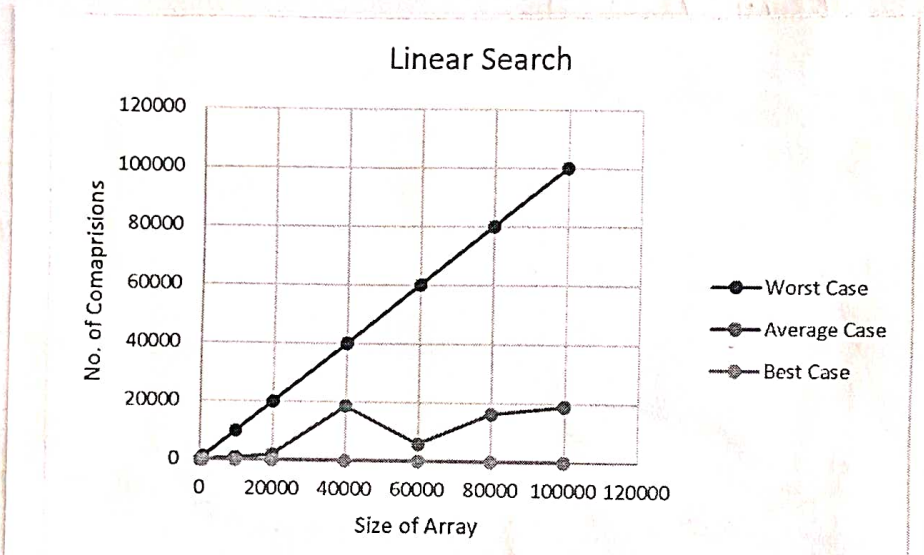
Date .....

Expt. No. ..... Page No. .....

```
void heapSort(int a[], int n){
    for(int n/2-1; i>=0; i--){
        heapify(a,n,i);
    }
    for(int i=n-1; i>0; i--){
        swap(a[0], a[i]);
        heapify(a,i,0);
    }
}

int main(){
    int size[] = {100, 1000, 10000, 20000, 30000, 60000, 80000, 100000};
    for(int j=0; j<8; j++){
        int n = size[j];
        int a[n+100];
        srand(time(0));
        for(int i=0; i<n; i++){
            a[i] = rand();
        }
        auto start = high_resolution_clock::now();
        heapSort(a,n);
        auto stop = high_resolution_clock::now();
        auto duration = duration_cast<microseconds>(stop - start);
        fstream file;
        file.open("output.txt", ios::out);
        for(int i=0; i<n; i++) file<<a[i]<<" ";
        file.close();
        cout<<duration.count()<<endl;
```

Teacher's Signature : _____


Linear Search
120000
100000
80000
60000
40000
20000
0
No. of Comaprisions
0
20000
40000
60000
80000
100000
120000
Size of Array
Worst Case
Average Case
Best Case

Date .....

Expt. No. ..... Page No. .....

Experiment 2.

Searching Linear Search and Binary Search

Program:

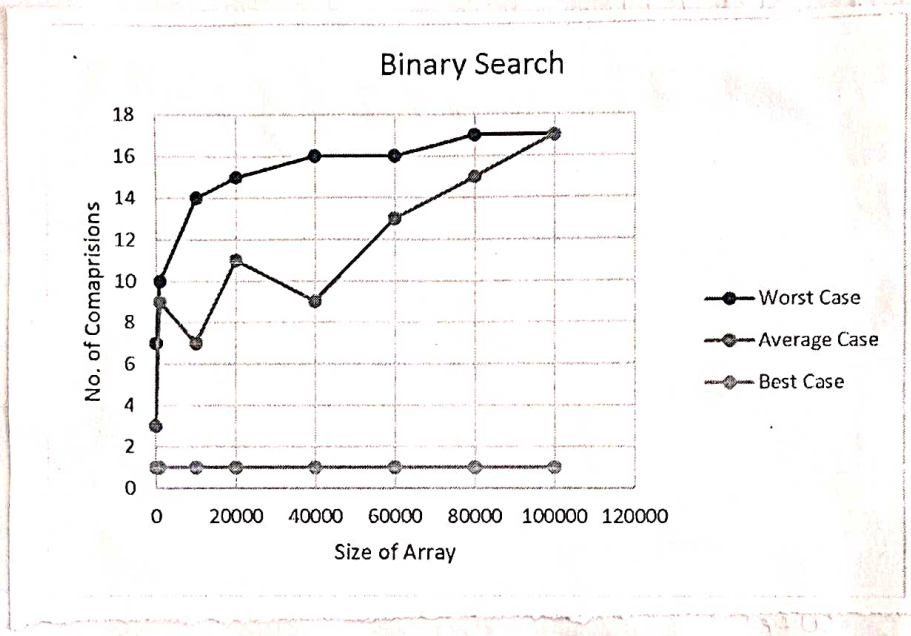
```
#include <bits/stdc++.h>
using namespace std;

int co=0;

int linearSearch(int a[], int n, int k){
    for(int i=0; i<n; i++){
        if(a[i]==k){
            return i;
        }
        co++;
    }
}

int binarySearch(int a[], int n, int k){
    int l=0, h=n-1;
    while(l<=h){
        int m = l + (h-l)/2;
        if(a[m]==k){
            co++;
```

Teacher's Signature : _____


Binary Search
No. of Comaprisions
18
16
14
12
10
8
6
4
2
0
0
20000
40000
60000
80000
100000
120000
Size of Array
Worst Case
Average Case
Best Case

Date .....

Expt. No. ..... Page No. .....

```
            return k;
        }else if(a[m]<k){
            l = m+1;
            co+=2;
        }else{
            h = m-1;}
        }
}

int main(){
    int size[] = {100, 1000, 10000, 20000, 40000, 60000, 80000, 100000};
    for(int j=0; j<8; j++){
        co=0;
        int n = size[j];
        int a[n+100];
        srand(time(0));
        for(int i=0; i<n; i++){
            a[i] = i; //best case.
            a[i] = n-i-1; //worst case;
            a[i] = rand(); //average case;
        {
        sort(a, a+n);
        linearSearch(a, n, a[rand()%n]); or binarySearch(a, n, a[rand()%n]);
        cout<<co<<endl;
    }
{
```

Teacher's Signature : _____

Date .....

Expt. No. ..... Page No. .....

Experiment 3 (a)

Merge and Quick Sort.

Program :-

```
#include <bits/stdc++.h>
#include <chrono>
using namespace std;
using namespace std::chrono;

void printarray(int a[], int n){
    fstream file;
    file.open("output.txt", ios::out);
    for(int i=0; i<n; i++){
        file << a[i] << " ";
    }
    file.close();
}

void swap(int *a, int *b){
    int temp = *a;
    *a = *b;
    *b = temp;
}
```

Teacher's Signature : _____

Date .....

Expt. No. ..... Page No. .....

```
void merge(int a[], int l, int m, int h){
    int n1 = m-l+1;
    int n2 = h-m;
    int i=0, j=0, k=l;
    int a1[n1], a2[n2];
    for(i=0; i<n1; i++) a1[i] = a[l+i];
    for(j=0; j<n2; j++) a2[j] = a[m+1+j];
    i=0;
    j=0)
    while(i<n1-1 && j<n2-1){
        if(a1[i] < a2[j]){
            a[k] = a1[i];
            i++;
            k++;
        }else{
            a[k] = a2[j];
            j++;
            k++;
        }
    }
    if(i!=n1){
        for(; i<n1; i++){
            a[k] = a1[i];
            k++)
        }
}
```

Teacher's Signature : _____

Date .....

Expt. No. ..... Page No. .....

```
    if(j!=n2){
        for(; j<n2; j++){
            a[k] = a2[j];
            k++;
        }
    }
}

void mergeSort(int a[], int l, int h){
    if(l<h){
        int m = (h+l)/2;
        mergeSort(a, l, m);
        mergeSort(a, m+1, h);
        merge(a, l, m, h);
    }
}

int partition(int a[], int l, int h){
    int p = a[l];
    int i=l, j=h;
    do{
        while(a[i]<=p){i++;}
        while(a[j]>p){j--;}
        if(i<j) swap(&a[i], &a[j]);
    }while(i<j);
    swap(&a[j], &a[l]);
    return j;
}
```

Teacher's Signature : _____

Date .....

Expt. No. ..... Page No. .....

```
void quickSort(int a[], int l, int h){
    if (l<h){
        int p = partition (a, l, h);
        quickSort (a, l, p-1);
        quickSort (a, p+1, h);
    }
    return;
}

int main(){
    int size[] = {100, 1000, 10000, 20000, 40000, 60000, 80000, 100000};
    for(int j=0; j<8; j++){
        int n = size[j];
        int a[n+100];
        srand(time(0));
        for(int i=0; i<n; i++){
            a[i] = i; or a[i] = n-i-1; or a[i] = rand();
        }
        auto start = high_resolution_clock::now();
        insertionSort(a,n); mergeSort(a,n); or quickSort(a,n);
        auto stop = high_resolution_clock::now();
        auto duration = duration_cast<microseconds>(stop-start);
        cout << "Time taken by function: " << duration.count() <<
            " microseconds" << endl;
```

Teacher's Signature : _____

Date .....

Expt. No. ..... Page No. .....

Experiment 3 (b)

Strassen's Matrix multiplication

Program:-

```
#include <bits/stdc++.h>
#include <chrono>
using namespace std;
Using namespace std::chrono;
typedef long long ll;

inline ll** MatrixMultiply(ll** a, ll **b, int n, int l, int m) {
    ll** c = new ll*[n];
    for(int i=0; i<n; i++)
        c[i] = new ll[m];

    for(int i=0; i<n; i++){
        for(int j=0; j<m; j++){
            c[i][j]=0;
            for(int k=0; k<l; k++){
                c[i][j] += a[i][k] * b[k][j];
            }
        }
    }
```

Teacher's Signature : _____

Date .....

Expt. No. ..... Page No. .....

```
    return c;
}

inline ull** Strassen(ull** a, ull** b, int n, int l, int m) {
    if (n==1 || l==1 || m==1)
        return MatrixMultiply(a,b,n,l,m);

    ull** c = new ull*[n];
    for(int i=0; i<n; i++){
        c[i] = new ull[m];

    int adjN = (n>>1) + (n&1);
    int adjL = (l>>1) + (l&1);
    int adjM = (m>>1) + (m&1);

    ull**** As = new ull***[2];
    for(int x=0; x<2; x++) {
        As[x] = new ull**[2];
        for(int y=0; y<2; y++){
            As[x][y] = new ull*[adjN];
            for(int i=0; i<adjN; i++){
                As[x][y][i] = new ull[adjL];
                for(int j=0; j<adjL; j++){
                    int I = i + (x&1)*adjN;
                    int J = j + (y&1)*adjL;
                    As[x][y][i][j] = (I<n && J<l) ? a[I][J] : 0;
```

Teacher's Signature : _____

Date .....

Expt. No. ..... Page No. .....

```
            }
        }
    }
}

Ud**** Bs = new Ud***[2];
for(int x=0; x<2; x++){
    Bs[x] = new Ud**[2];
    for(int y=0; y<2; y++){
        Bs[x][y] = new Ud*[adjN];
        for(int i=0; i<adjL; i++){
            Bs[x][y][i] = new Ud[adjM];
            for(int j=0; j<adjM; j++){
                int I = i+(x&1)*adjL;
                int J = j+(y&1)*adjM;
                Bs[x][y][i][j] = (I<l && J<m)? b[I][J] : 0;
            }
        }
    }
}

Ud*** s = new Ud**[10];
for(int i=0; i<10; i++){
    switch(i){
    case 0:
        s[i] = new Ud*[adjL];
```

Teacher's Signature : _____

Date .....

Expt. No. ..... Page No. .....

```
        for(int j=0; j<adjL; j++){
            s[i][j] = new ld [adjM];
            for(int k=0; k<adjM; k++){
                s[i][j][k] = Bs[0][1][j][k] - Bs[1][1][j][k];
            }
        }
        break;
    case 1:
        s[i] = new ld* [adjN];
        for(int j=0; j<adjN; j++){
            s[i][j] = new ld [adjL];
            for(int k=0; k<adjL; k++){
                s[i][j][k] = As[0][0][j][k] + As[0][1][j][k];
            }
        }
        break;
    case 2:
        s[i] = new ld* [adjN];
        for(int j=0; j<adjN; j++){
            s[i][j] = new ld [adjL];
            for(int k=0; k<adjL; k++){
                s[i][j][k] = As[1][0][j][k] + As[1][1][j][k];
            }
        }
        break;
```

Teacher's Signature : _____

Date .....

Expt. No. ..... Page No. .....

```
case 3;
    s[i] = new ull* [adjL];
    for(int j=0; j< adjL; j++){
        s[i][j] = new ull [adjM];
        for(int k=0; k<adjM; k++){
            s[i][j][k] = Bs[1][0][j][k] - Bs[0][0][j][k];
        }
    }
    break;
case 4;
    s[i] = new ull* [adjN];
    for (int j=0; j<adjN; j++){
        s[i][j] = new ull [adjL];
        for(int k=0; k<adjL; k++){
            s[i][j][k] = As[0][0][j][k] + As[1][1][j][k];
        }
    }
    break;
case 5;
    s[i] = new ull* [adjL];
    for(int j=0; j<adjL; j++){
        s[i][j] = new ull[adjM];
        for(int k=0; k<adjM; k++){
            s[i][j][k] = Bs[0][0][j][k] + Bs[1][1][j][k];
        }
    }
    break;
```

Teacher's Signature : _____

Date .....

Expt. No. ..... Page No. .....

```
case 6:
    s[i] = new ld*[adjN];
    for(int j=0; j<adjN; j++){
        s[i][j] = new ld[adjL];
        for(int k=0; k<adjL; k++)
            s[i][j][k] = As[0][i][j][k] - As[1][i][j][k];
    }
}
break;
case 7:
    s[i] = new ld*[adjL];
    for(int j=0; j<adjL; j++){
        s[i][j] = new ld[adjM];
        for(int k=0; k<adjM; k++){
            s[i][j][k] = Bs[1][0][j][k] + Bs[1][i][j][k];
        }
    }
}
break;
case 8:
    s[i] = new ld*[adjN];
    for(int j=0; j<adjN; j++){
        s[i][j] = new ld[adjL];
        for(int k=0; k<adjL; k++){
            s[i][j][k] = As[0][0][j][k] - As[1][0][j][k];
        }
    }
}
break;
```

Teacher's Signature : _____

Date .....

Expt. No. ..... Page No. .....

```
        case 9:
            S[i] = new Ud* [adjL];
            for(int j=0; j<adjL; j++){
              S[i][j] = new Ud [adjM];
              for(int k=0; k<adjM; k++){
                S[i][j][k] = Bs[0][0][j][k] + Bs[0][1][j][k];
              }
            }
        break;
      }
    }

Ud*** p = new Ud** [7];
p[0] = Strassen(As[0][0], S[0], adjN, adjL, adjM);
p[1] = Strassen(S[1], Bs[1][1], adjN, adjL, adjM);
p[2] = Strassen(S[2], Bs[0][0], adjN, adjL, adjM);
p[3] = Strassen(As[1][1], S[3], adjN, adjL, adjM);
p[4] = Strassen(S[4], S[5], adjN, adjL, adjM);
p[5] = Strassen(S[6], S[7], adjN, adjL, adjM);
p[6] = Strassen(S[8], S[9], adjN, adjL, adjM);

for(int i=0; i<adjN; i++){
  for(int j=0; j<adjM; j++){
    C[i][j] = p[4][i][j] + p[3][i][j] - p[1][i][j] + p[5][i][j];
    if(j+adjM < m)
      C[i][j+adjM] = p[0][i][j] + p[1][i][j];
```

Teacher's Signature : _____

Date .....

Expt. No. ..... Page No. .....

```
        if (i+adjN < n){
            c[i+adjN][j] = p[2][i][j] + p[3][i][j];
        if (i+adjN<n && j+adjM<m)
            c[i+adjN][j+adjM] = p[4][i][j] + p[6][i][j] - p[2][i][j]
                                - p[6][i][j])
    }
}

for( int x = 0; x<2; x++){
    for (int y=0; y<2; y++){
        for (int i=0; i<adjN; i++)
            delete[] As[x][y][i];
        }
        delete[] As[x][y];
    }
    delete[] As[x];
}
delete[] As;

for(int x=0; x<2; x++){
    for(int y=0; y<2; y++){
        for(int i=0; i<adjL; i++){
            delete Bs[x][y][i];
        }
        delete[]Bs[x][y];
    }
    delete[] Bs[x];
}
delete[] Bs;
```

Teacher's Signature : _____

Date .....

Expt. No. ..... Page No. .....

```
for (int i=0; i<10; i++) {
    switch (i) {
    case 0:
    case 3:
    case 5:
    case 7:
    case 9:
        for (int j=0; j<adj[i]; j++) {
            delete[] s[i][j];
        }
        break;
    case 1:
    case 2:
    case 4:
    case 6:
    case 8:
        for (int j=0; j<adjN; j++) {
            delete[] s[i][j];
        }
        break;
    }
    delete[] s[i];
}
delete[] s;

for(int i=0; i<7; i++) {
```

Teacher's Signature : _____

Date .....

Expt. No. ..... Page No. .....

```
    for (int j=0; j<(n>>1); j++){
        delete [] p[i][j];
    }
    delete [] p[i];
}
delete [] p;

return C;
}

int main(){
    ull** matA;
    matA = new ull*[2];
    for(int i=0; i<2; i++)
        matA[i] = new ull[3];
    srand(time(0));
    for(int i=0; i<2; i++){
        for(int j=0; j<3; j++){
            matA[i][j] = rand();
        }
    }
    ull** matB;
    matB = new ull*[3];
    for(int i=0; i<3; i++)
        matB[i] = new ull[2];
```

Teacher's Signature : .....

Date .....

Expt. No. ..... Page No. .....

```
for(int i=0; i<3; i++){
    for(int j=0; j<2; j++){
        matB[i][j] = rand();

ll** matC = Strassen(matA, matB, 2, 3, 2);
for (int i=0; i<2; i++){
    for (int j=0; j<2; j++){
        printf("%lld", matC[i][j]);
    }
    printf("\n");
}
return 0;
}
```

Teacher's Signature :.....

Date .....

Expt. No. ..... Page No. .....

Experiment No- 9

Sorting in linear time : Counting Sort, Radix Sort.

```
# include <bits/stdc++.h>
# include < chrono >
using namespace std;
using namespace std::chrono;

void printarray(int a[], int n){
    fstream file;
    file.open("Output.txt", ios::out);
    for( int i=0; i<n; i++){
        file << a[i] << " ";
    }
    file.close()
}

void countSort(int a[], int n) {
    int maxe = a[0];
    for( int i=0; i<n; i++) {
        maxe = max(maxe, a[i]);
    }
    int h[maxe+1] = {0};
    for( int j=0; j<n) j++){
        h[a[j]]++;
    }
    int i=0
```

Teacher's Signature : _____

Date .....

Expt. No. ..... Page No. .....

```
for (int K=0; K<maxe; K++){
        while (h[K] >0){
            a[i] = K;
            i++;
            h[K]--;
        }
    }
    printarray(a,n);
    return 0;
}

void radixSort( char a[], int n){
    int h[26] = {0};
    for(int i=0; i<n; i++){
        h[a[i]-'a']++;
    }
    for( int i=0; i<n; i++){
        while(h[i] >0){
            a[i] = i+'a';
        }
    }
    printArray(a,n);
}

int main(){
    int n=100000;
    int a[n+100];
```

Teacher's Signature : _____

Date .....

Expt. No. ..... Page No. .....

```
srand(time(0));
for(int i=0; i<n; i++){
    a[i] = i;
}
auto start = high_resolution_clock::now();
countingSort(a,n); or radixSort(a,n);
auto stop = high_resolution_clock::now();
auto duration = duration_cast<microseconds>(stop-start);
cout<<"Time taken by function: "<<duration.count()<<
"microseconds"<<endl;
}
```

Teacher's Signature : _____

Date .....

Expt. No. ..... Page No. .....

Experiment 5

Maximum / Minimum element, K^{th} Smallest Element :-

Program :-

```
#include <bits/stdc++.h>
using namespace std;

Struct Pair
{
    int min;
    int max;
}

Struct Pair getMinMax(int arr[], int low, int high){
    struct Pair minmax, mml, mmr;
    int mid;

    if (low == high)
    {
        minmax.max = arr[low];
        minmax.min = arr[low];
        return minmax;
    }

    if (high == low + 1) {
```

Date .....

Expt. No. ..... Page No. .....

```
        if (arr[low] > arr[high])
        {
            minmax.max = arr[low];
            minmax.min = arr[high];
        }else{
            minmax.max = arr[high];
            minmax.min = arr[low];
        }
        return minmax;
    }
    mid = (low + high)/2;
    mml = getMinMax(arr, low, mid);
    mmr = getMinMax(arr, mid+1, high);

    if(mml.min < mmr.min)
        minmax.min = mml.min;
    else
        minmax.min = mmr.min;

    if (mml.max > mmr.max)
        minmax.max = mml.max;
    else
        minmax.max = mmr.max;

    return minmax;
}
```

Teacher's Signature : _____

Date

Expt. No. ..... Page No. .....

```
int KthSmallest(int arr[], int n, int K){

    sort(arr, arr+n);
    return arr[K-1];
}

int main(){
    int n = 100000;
    int a[n+100];
    srand(time(0));
    for(int i=0; i<n; i++){
        a[i] = i;  or  a[i] = n-i-1;  or  a[i] = rand();
    }

    auto start = high_resolution_clock::now();
    getMinMax(a, n);
    auto stop = high_resolution_clock::now();
    auto duration = duration_cast<microseconds>(stop - start);
    cout << "Time taken by function: " << duration.count() <<
     "microseconds" << endl;
}
```

Teacher's Signature : _____

Date

Expt. No. Page No.

Experiment 6

Minimum Spanning Tree using Prim's and Kruskal's Algorithm.

Program:-

```
int minKey(int key[], bool mstSet[]) {
    int min = INT_MAX, min_index;
    for (int v = 0; v < 5; v++) {
        if (mstSet[v] == false && key[v] < min)
            min = key[v];
            min_index = v;
    }
    return min_index;
}

void printMST(int parent[], int graph[5][5]){
    cout << "Edge \t weight \n";
    for(int i=1; i<V; i++)
        cout<<parent[i]<<" - "<<i<<" \t"<<graph[i][parent[i]]<<
        "\n";
}

void primMST( int graph[5][5]) {
    int parent[5];
    int key[5])
```

Teacher's Signature :

Date .....

Expt. No. ..... Page No. .....

```
bool mstSet[V];
for (int i=0; i<5; i++)
    Key[i] = INT_MAX, mstSet[i] = false;
key[0] = 0;
parent[0] = -1;
for(int count = 0; count < V-1; count++)
{
    int u = minKey(Key, mstSet);
    mstSet[u] = true;
    for (int v=0; v<5; v++)
        if (graph[u][v] && mstSet[v] == false && graph[u][v]
                                                < key[v])
            parent[v] = u, key[v] = graph[u][v];
}
printMST(parent, graph);
}

// Kruskal Algorithm.

class DSU{
    int *parent;
    int *rank;
public:
    DSU(int n){
        parent = new int[n];
        rank = new int[n];
```

Teacher's Signature : _____

Date .....

Expt. No. ..... Page No. .....

```
for(int i=0; i<n; i++){
    parent[i] = -1;
    rank[i] = 1; }
}

int find(int i){
    if(parent[i] == -1)
        return i;
    return parent[i] = find(parent[i]);
}

void unite(int x, int y){
    int s1 = find(x);
    int s2 = find(y);

    if(s1 != s2){
        if(rank[s1] < rank[s2]){
            parent[s2] = s2;
            rank[s2] += rank[s1];
        }else{
            parent[s2] = s1;
            rank[s1] += rank[s2]; }
    }

class Graph {
    vector<vector<int>> edgelist;
    int V;
```

Teacher's Signature : _____

```
public:
    Graph(int v){
        this -> V = v;
    }
    void addEdge(int x, int y, int w){
        edgelist.push_back({w,x,y});
    }
    int kruskals_mst(){
        sort(edgelist.begin(), edgelist.end());
        DSU s(v);
        int ans = 0;
        for(auto edge : edgelist)
        {
            int w = edge[0];
            int x = edge[1];
            int y = edge[2];
            if (s.find(x) != s.find(y)){
                s.unite(x,y);
                ans += w;
            }
        }
    }   return ans;
};
```

Date .....

Expt. No. .....

Page No. .....

Experiment 7

Dynamic Programming: Matrix Chain Multiplication:

```
int matrixChainOrder(int p[], int n)
    int dp[n][n];
    for(int i=1; i<n; i++){
        dp[i][i]=0
    for(int L=1; L<n-1; L++){
        for(int i=1; i<n-L; i++){
            dp[i][i+L] = min(dp[i+1][i+L] + p[i-1]*p[i]*p[i+L],
            dp[i][i+L-1] + p[i-1]*p[i+L-1]*p[i+L]);
        }
    }
    return dp[1][n-1];
}
```

Longest Common Subsequence:-

```
int lcs( char *X, char *Y, int m, int n) {
    int L[m+1][n+1];
    int i, j;
    for( i=0; i<=m; i++){
        for( j=0; j<=n ) j++) {
            if(i==0 || j==0) {
                L[i][j] = 0;
```

Teacher's Signature : _____

```
        else if (X[i-1] == Y[j-1])
            L[i][j] = L[i-1][j-1] + 1;
        else
            L[i][j] = max(L[i-1][j], L[i][j-1]);
        }
    }
    return L[m][n];
}
```

Date .....

Expt. No. ..... Page No. .....

Experiment 8

Case study of P, NP, NP- complete and NP Hard Problems.

In computer science, the problem are basically divided into the following parts -

Optimization Problems :- Optimization problem are those for which the objective is to maximize or minimize some values.

Decision Problems :- There are many problems for which the answer is a Yes or No. These kind of problems are known as decision problem.

Below are the major classes of the problems :-

P - class :- The class P consist of those problems that are solvable in polynomial time i.e. these problems can be solved in time $O(n^k)$ in worst case where k is a constant.

Example :- "String Matching Problem".
We are given two string S and T (& Length(S) > Length(T)) and we need to check is any substring of S matches with T or not.

We can solve this problem by iterating in S and if any character matches with the first letter of T match the next letter until the string matched or any unmatched character found.

Teacher's Signature : _____

Date .....

Expt. No. ..... Page No. .....

Time Complexity = $O(m.n)$ is the worst case.

NP class:- The class consist of those problems that are verifiable in polynomial time. Every problem in this class can be solved in exponential time using exhaustive search.
Eg:- "Travelling Salesman problem".

The decision version of travelling salesman problem is NP. The verification can be clearly be done in is polynomial time. It simply adds the matrix entries ~~corresponding~~ corresponding to the paths between the cities.

NP Complete Problems:- In computational complexity theory, a problem is NP complete when -

1) A non deterministic Turing Machine can solve it in polynomial time.
2) A deterministic Turing Machine can solve it in a large time complexity.
3) It can be used to simulate any other problem with similar solvability.

Eg. "Knapsack problem"

NP Hard Problems:- In computational theory, NP hardness (non-deterministic polynomial time hardness) is the defining property of a class of problems that are informally "at least as hard as the hardest problem in NP."

Teacher's Signature : _____

Date .....

Expt. No. ..... Page No. .....

Example:- "Longest Path Problem"

Given an undirected graph G and two vertices u and v, find the longest simple path from u to v.

Teacher's Signature : _____