Practical = 5

Name :shubham shivraj Suryawanshi
Reg no 2020BIT004

Write a C/C++ Code to implement (With Practical example Implementation)
1) Merge Sort
2) Binary Search
3) Quick Sort
4) Strassen's Matrix multiplication

1) Merge Sort

Code:-

```
#include <stdio.h>
#include <stdlib.h>
void merge(int arr[], int l,
                int m, int r)
{         int i, j, k;
          int n1 = m - l + 1;
          int n2 = r - m;
          int L[n1], R[n2];
          for (i = 0; i < n1; i++)
                L[i] = arr[l + i];
          for (j = 0; j < n2; j++)
                R[j] = arr[m + 1 + j];
          i = 0;
          j = 0;
          k = l;
          while (i < n1 && j < n2)
          {                   if (L[i] <= R[j])
                {arr[k] = L[i];
                          i++;}
                else
                {arr[k] = R[j];
                          j++;
                }
                k++;}
          while (i < n1) {
                arr[k] = L[i];
                i++;
                k++;}
          while (j < n2)
          {arr[k] = R[j];
                j++;
                k++;
          }
}
void mergeSort(int arr[],
                          int l, int r)
{         if (l < r)
          {                   int m = l + (r - l) / 2;
                mergeSort(arr, l, m);
                mergeSort(arr, m + 1, r);
                merge(arr, l, m, r);
          }
}
void printArray(int A[], int size)
{         int i;
          for (i = 0; i < size; i++)
                printf("%d ", A[i]);
          printf("\n");
}
int main()
```

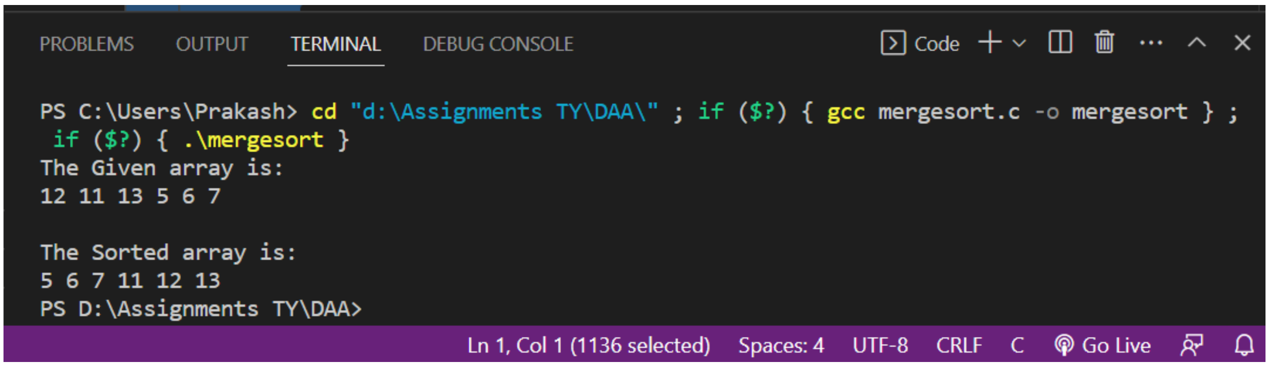
```
        {
                int arr[] = {12, 11, 13, 5, 6, 7};
                int arr_size = sizeof(arr) / sizeof(arr[0]);

                printf("The Given array is: \n");
                printArray(arr, arr_size);

                mergeSort(arr, 0, arr_size - 1);

                printf("\nThe Sorted array is: \n");
                printArray(arr, arr_size);
                return 0;
}
```

Output:



```
PS C:\Users\Prakash> cd "d:\Assignments TY\DAA\" ; if ($?) { gcc mergesort.c -o mergesort } ;
 if ($?) { .\mergesort }
The Given array is:
12 11 13 5 6 7

The Sorted array is:
5 6 7 11 12 13
PS D:\Assignments TY\DAA>
```

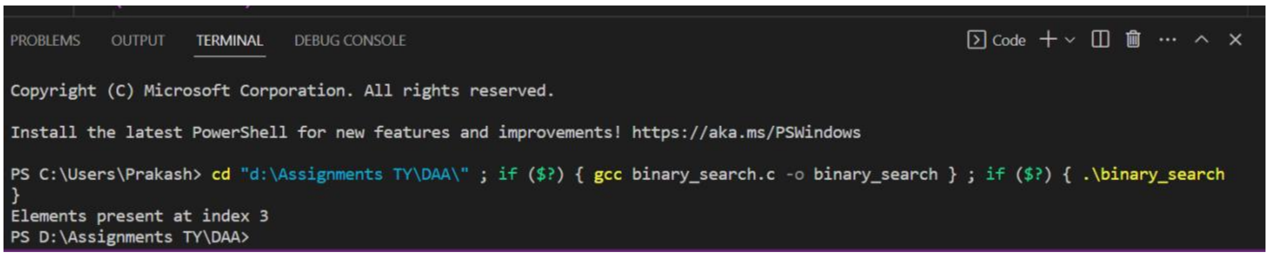
✓ 2) Binary Search

Code:-

```
#include <stdio.h>
int binarySearch(int arr[], int l, int r, int x)
{     if (r >= l) {
            int mid = l + (r - l) / 2;
            if (arr[mid] == x)
                  return mid;
            if (arr[mid] > x)
                  return binarySearch(arr, l, mid - 1, x);
            return binarySearch(arr, mid + 1, r, x);
      }
      return -1;
}

int main(void)
{     int arr[] = { 2, 3, 4, 10, 40 };
      int n = sizeof(arr) / sizeof(arr[0]);
      int x = 10;
      int result = binarySearch(arr, 0, n - 1, x);
      (result == -1)
            ? printf("Elements not in array")
            : printf("Elements present at index %d", result);
      return 0;
      }
```

Output:



```
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Prakash> cd "d:\Assignments TY\DAA\" ; if ($?) { gcc binary_search.c -o binary_search } ; if ($?) { .\binary_search
}
Elements present at index 3
PS D:\Assignments TY\DAA>
```

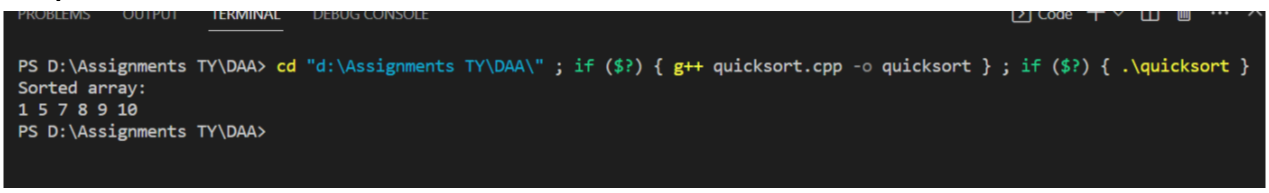
✓ 3) Quick Sort

Code:-

```
#include <bits/stdc++.h>
using namespace std;
void swap(int* a, int* b)
{
        int t = *a;
        *a = *b;
        *b = t;
}
int partition(int arr[], int low, int high)
{       int pivot = arr[high]; // pivot
        int i
                = (low
                - 1);
        for (int j = low; j <= high - 1; j++) {
                if (arr[j] < pivot) {
                        i++;
                        swap(&arr[i], &arr[j]);
                }
        }
        swap(&arr[i + 1], &arr[high]);
        return (i + 1);
}
void quickSort(int arr[], int low, int high)
{       if (low < high) {
                int pi = partition(arr, low, high);
                quickSort(arr, low, pi - 1);
                quickSort(arr, pi + 1, high);
        }
}
void printArray(int arr[], int size)
{
        int i;
        for (i = 0; i < size; i++)
                cout << arr[i] << " ";
        cout << endl;
}
int main()
{       int arr[] = { 10, 7, 8, 9, 1, 5 };
        int n = sizeof(arr) / sizeof(arr[0]);
        quickSort(arr, 0, n - 1);
        cout << "Sorted array: \n";
        printArray(arr, n);
```

```
                return 0;
}
```

Output:-



```
PS D:\Assignments TY\DAA> cd "d:\Assignments TY\DAA\" ; if ($?) { g++ quicksort.cpp -o quicksort } ; if ($?) { .\quicksort }
Sorted array:
1 5 7 8 9 10
PS D:\Assignments TY\DAA>
```

✓ 4) Strassen's Matrix multiplication

Code:-

```
#include <bits/stdc++.h>
using namespace std;
typedef long long lld;
inline lld** MatrixMultiply(lld** a, lld** b, int n,
                                                                int l, int m)
{           lld** c = new lld*[n];
 for (int i = 0; i < n; i++)
            c[i] = new lld[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];
                        }
            }
 }
 return c;
}
inline lld** Strassen(lld** a, lld** b, int n,
                                                        int l, int m)
{           if (n == 1 || l == 1 || m == 1)
            return MatrixMultiply(a, b, n, l, m);
 lld** c = new lld*[n];
 for (int i = 0; i < n; i++)
            c[i] = new lld[m];
 int adjN = (n >> 1) + (n & 1);
 int adjL = (l >> 1) + (l & 1);
int adjM = (m >> 1) + (m & 1);
lld**** As = new lld***[2];
for (int x = 0; x < 2; x++) {
        As[x] = new lld**[2];
        for (int y = 0; y < 2; y++) {
                As[x][y] = new lld*[adjN];
                for (int i = 0; i < adjN; i++) {
                        As[x][y][i] = new lld[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;
                        }
                }
        }
}
lld**** Bs = new lld***[2];
for (int x = 0; x < 2; x++) {
        Bs[x] = new lld**[2];
        for (int y = 0; y < 2; y++) {
                Bs[x][y] = new lld*[adjN];
                for (int i = 0; i < adjL; i++) {
                        Bs[x][y][i] = new lld[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;
                        }
                }
        }
}
lld*** s = new lld**[10];
for (int i = 0; i < 10; i++) {
        switch (i) {
        case 0:
```

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

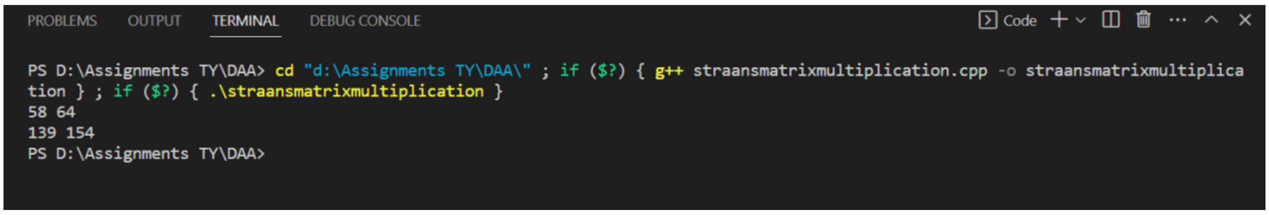
```
case 4:
        s[i] = new lld*[adjN];
        for (int j = 0; j < adjN; j++) {
                s[i][j] = new lld[adjL];
                for (int k = 0; k < adjL; k++) {
                        s[i][j][k] = As[0][0][j][k] + As[1][1][j][k];
                }
        }
        break;
case 5:
        s[i] = new lld*[adjL];
        for (int j = 0; j < adjL; j++) {
                s[i][j] = new lld[adjM];
                for (int k = 0; k < adjM; k++) {
                        s[i][j][k] = Bs[0][0][j][k] + Bs[1][1][j][k];
                }
        }
        break;
case 6:
        s[i] = new lld*[adjN];
        for (int j = 0; j < adjN; j++) {
                s[i][j] = new lld[adjL];
                for (int k = 0; k < adjL; k++) {
                        s[i][j][k] = As[0][1][j][k] - As[1][1][j][k];
                }
        }
        break;
case 7:
        s[i] = new lld*[adjL];
        for (int j = 0; j < adjL; j++) {
                s[i][j] = new lld[adjM];
                for (int k = 0; k < adjM; k++) {
                        s[i][j][k] = Bs[1][0][j][k] + Bs[1][1][j][k];
                }
        }
```

```
                break;
            case 8:
                s[i] = new lld*[adjN];
                for (int j = 0; j < adjN; j++) {
                    s[i][j] = new lld[adjL];
                    for (int k = 0; k < adjL; k++) {
                        s[i][j][k] = As[0][0][j][k] - As[1][0][j][k];
                    }
                }
                break;
            case 9:
                s[i] = new lld*[adjL];
                for (int j = 0; j < adjL; j++) {
                    s[i][j] = new lld[adjM];
                    for (int k = 0; k < adjM; k++) {
                        s[i][j][k] = Bs[0][0][j][k] + Bs[0][1][j][k];
                    }
                }
                break;
            }
        }
        lld*** p = new lld**[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];
                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[0][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;
for (int i = 0; i < 10; i++) {
        switch (i) {
        case 0:
        case 3:
        case 5:
        case 7:
        case 9:
                for (int j = 0; j < adjL; 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++) {
            for (int j = 0; j < (n >> 1); j++) {
                delete[] p[i][j];
            }
            delete[] p[i];
    }
    delete[] p;
    return c;
}
int main(){
    lld** matA;
    matA = new lld*[2];
    for (int i = 0; i < 2; i++)
            matA[i] = new lld[3];
    matA[0][0] = 1;
    matA[0][1] = 2;
    matA[0][2] = 3;
    matA[1][0] = 4;
    matA[1][1] = 5;
    matA[1][2] = 6;
    lld** matB;
```

```
matB = new lld*[3];
for (int i = 0; i < 3; i++)
matB[i] = new lld[2];
matB[0][0] = 7;
matB[0][1] = 8;
matB[1][0] = 9;
matB[1][1] = 10;
matB[2][0] = 11;
matB[2][1] = 12;
lld** 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;
```

Output:-



```
PS D:\Assignments TY\DAA> cd "d:\Assignments TY\DAA\" ; if ($?) { g++ straansmatrixmultiplication.cpp -o straansmatrixmultiplica
tion } ; if ($?) { .\straansmatrixmultiplication }
58 64
139 154
PS D:\Assignments TY\DAA>
```