✅ SET A

1. Fibonacci Series

```
#include <stdio.h>

int main() {
    int n, a = 0, b = 1, c;
    printf("Enter number of terms: ");
    scanf("%d", &n);

    printf("Fibonacci Series: ");
    for(int i = 1; i <= n; i++) {
        printf("%d ", a);
        c = a + b;
        a = b;
        b = c;
    }
    return 0;
}
```

2. Read & Display Marks

```
#include <stdio.h>

int main() {
    int n;
    printf("Enter number of students: ");
    scanf("%d", &n);

    int marks[n];
    for(int i=0; i<n; i++) {
        printf("Enter marks of student %d: ", i+1);
        scanf("%d", &marks[i]);
    }

    printf("\nMarks of Students:\n");
    for(int i=0; i<n; i++)
        printf("Student %d = %d\n", i+1, marks[i]);

    return 0;
}
```

✅ SET B

1. Compare 3 numbers – Min & Max

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

```
int main() {
    int a, b, c;
    printf("Enter three numbers: ");
    scanf("%d %d %d", &a, &b, &c);

    int max = a, min = a;

    if(b > max) max = b;
    if(c > max) max = c;

    if(b < min) min = b;
    if(c < min) min = c;

    printf("Maximum = %d\nMinimum = %d\n", max, min);
    return 0;
}
```

2. Matrix Multiplication

```
#include <stdio.h>
int main() {
    int a[10][10], b[10][10], c[10][10];
    int r1, c1, r2, c2;

    printf("Enter rows & cols of Matrix 1: ");
    scanf("%d %d", &r1, &c1);
    printf("Enter rows & cols of Matrix 2: ");
    scanf("%d %d", &r2, &c2);

    if(c1 != r2) {
        printf("Multiplication Not Possible!");
        return 0;
    }

    printf("Enter Matrix 1:\n");
    for(int i=0;i<r1;i++)
        for(int j=0;j<c1;j++)
            scanf("%d",&a[i][j]);

    printf("Enter Matrix 2:\n");
    for(int i=0;i<r2;i++)
```

```
    for(int j=0;j<c2;j++)
        scanf("%d",&b[i][j]);

  // multiplication
  for(int i=0;i<r1;i++)
    for(int j=0;j<c2;j++) {
      c[i][j] = 0;
      for(int k=0;k<c1;k++)
        c[i][j] += a[i][k] * b[k][j];
    }

  printf("Result Matrix:\n");
  for(int i=0;i<r1;i++) {
    for(int j=0;j<c2;j++)
      printf("%d ", c[i][j]);
    printf("\n");
  }
  return 0;
}
```

✅ SET C

1D Array – Sum

```
#include <stdio.h>
int main() {
  int n, sum = 0;
  printf("Enter size: ");
  scanf("%d", &n);

  int a[n];
  for(int i=0; i<n; i++) {
    printf("Enter element %d: ", i+1);
    scanf("%d", &a[i]);
    sum += a[i];
  }

  printf("Sum of array = %d", sum);
  return 0;
}
```

Add Two Matrices

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

```
int main() {
   int r, c, a[10][10], b[10][10], s[10][10];

   printf("Enter rows & columns: ");
   scanf("%d %d", &r, &c);

   printf("Enter Matrix 1:\n");
   for(int i=0;i<r;i++)
      for(int j=0;j<c;j++)
         scanf("%d",&a[i][j]);

   printf("Enter Matrix 2:\n");
   for(int i=0;i<r;i++)
      for(int j=0;j<c;j++)
         scanf("%d",&b[i][j]);

   for(int i=0;i<r;i++)
      for(int j=0;j<c;j++)
         s[i][j] = a[i][j] + b[i][j];

   printf("Sum Matrix:\n");
   for(int i=0;i<r;i++) {
      for(int j=0;j<c;j++)
         printf("%d ", s[i][j]);
      printf("\n");
   }
   return 0;
}
```

✅ SET D

1. Bubble Sort (Age Sorting)

```
#include <stdio.h>
int main() {
   int n;
   printf("Enter number of ages: ");
   scanf("%d",&n);

   int a[n];
   for(int i=0;i<n;i++)
      scanf("%d",&a[i]);
```

```
    for(int i=0;i<n-1;i++)
        for(int j=0;j<n-i-1;j++)
            if(a[j] > a[j+1]) {
                int temp = a[j];
                a[j] = a[j+1];
                a[j+1] = temp;
            }

    printf("Sorted Ages:\n");
    for(int i=0;i<n;i++)
        printf("%d ", a[i]);

    return 0;
}
```

2. Singly Linked List – Delete PRN

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

struct node {
    int prn;
    struct node *next;
};

int main() {
    struct node *head=NULL, *temp, *p;
    int n, del;

    printf("Enter number of students: ");
    scanf("%d",&n);

    for(int i=0;i<n;i++) {
        temp = (struct node*)malloc(sizeof(struct node));
        printf("Enter PRN: ");
        scanf("%d",&temp->prn);
        temp->next = head;
        head = temp;
    }
```

```
    printf("Enter PRN to delete: ");
    scanf("%d",&del);

    temp = head;
    p = NULL;

    while(temp != NULL && temp->prn != del) {
        p = temp;
        temp = temp->next;
    }

    if(temp == NULL)
        printf("PRN not found!\n");
    else {
        if(p == NULL) head = temp->next;
        else p->next = temp->next;
        free(temp);
        printf("PRN deleted.\n");
    }

    return 0;
}
```

✅ SET E

1. Quick Sort (Age Sorting)

```
#include <stdio.h>

void quicksort(int a[], int low, int high) {
    if(low < high) {
        int pivot = a[low];
        int i = low, j = high, temp;

        while(i < j) {
            while(a[i] <= pivot && i <= high) i++;
            while(a[j] > pivot) j--;
            if(i < j) {
                temp = a[i]; a[i] = a[j]; a[j] = temp;
            }
```

```
        }

        a[low] = a[j];
        a[j] = pivot;

        quicksort(a, low, j-1);
        quicksort(a, j+1, high);
    }
}

int main() {
    int n;
    printf("Enter number of ages: ");
    scanf("%d",&n);

    int a[n];
    for(int i=0; i<n; i++)
        scanf("%d",&a[i]);

    quicksort(a,0,n-1);

    printf("Sorted Ages:\n");
    for(int i=0; i<n; i++)
        printf("%d ",a[i]);

    return 0;
}
```

2. Queue – Enqueue & Dequeue

```
#include <stdio.h>
#define SIZE 5

int queue[SIZE], front = -1, rear = -1;

void enqueue(int x) {
    if(rear == SIZE-1)
        printf("Queue Overflow\n");
    else {
        if(front == -1) front = 0;
        queue[++rear] = x;
```

```
    }
}

void dequeue() {
    if(front == -1)
        printf("Queue Underflow\n");
    else {
        printf("Deleted: %d\n", queue[front]);
        if(front == rear)
            front = rear = -1;
        else
            front++;
    }
}

int main() {
    enqueue(10);
    enqueue(20);
    enqueue(30);

    dequeue();
    dequeue();

    return 0;
}
```

✅ SET F

1. Linear Search – CGPA = 9.5

```
#include <stdio.h>
int main() {
    int n, roll = -1;
    printf("Enter number of students: ");
    scanf("%d",&n);

    float cgpa[n];
    for(int i=0;i<n;i++) {
        printf("Enter CGPA of roll %d: ", i+1);
        scanf("%f",&cgpa[i]);
    }
```

```
    for(int i=0;i<n;i++)
        if(cgpa[i] == 9.5)
            roll = i+1;

    if(roll != -1)
        printf("Student with CGPA 9.5 is roll no: %d", roll);
    else
        printf("No student with CGPA 9.5");

    return 0;
}
```

2. Binary Tree Implementation

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

struct node {
    int data;
    struct node *left, *right;
};

struct node* create(int x) {
    struct node* new = (struct node*)malloc(sizeof(struct node));
    new->data = x;
    new->left = new->right = NULL;
    return new;
}

struct node* insert(struct node* root, int x) {
    if(root == NULL)
        return create(x);

    if(x < root->data)
        root->left = insert(root->left, x);
    else
        root->right = insert(root->right, x);

    return root;
}
```

```
void inorder(struct node* root) {
   if(root != NULL) {
      inorder(root->left);
      printf("%d ", root->data);
      inorder(root->right);
   }
}

int main() {
   struct node *root = NULL;
   root = insert(root, 50);
   insert(root, 30);
   insert(root, 70);
   insert(root, 20);
   insert(root, 40);

   printf("Inorder Traversal: ");
   inorder(root);

   return 0;
}
```

✅ SET G

1. Binary Search – CGPA = 6.0

```
#include <stdio.h>

int main() {
   int n;
   printf("Enter number of students: ");
   scanf("%d",&n);

   float cgpa[n];
   printf("Enter CGPAs (sorted):\n");
   for(int i=0;i<n;i++)
      scanf("%f",&cgpa[i]);

   float key = 6.0;
   int low=0, high=n-1, mid, roll=-1;

   while(low <= high) {
```

```
        mid = (low+high)/2;
        if(cgpa[mid] == key) {
            roll = mid+1;
            break;
        }
        else if(cgpa[mid] < key)
            low = mid + 1;
        else
            high = mid - 1;
    }

    if(roll != -1)
        printf("Roll No with CGPA 6.0 = %d", roll);
    else
        printf("CGPA 6.0 not found");

    return 0;
}
```

2. Kruskal MST

```
#include <stdio.h>

int find(int parent[], int i) {
    while(parent[i] != i)
        i = parent[i];
    return i;
}

void union_set(int parent[], int x, int y) {
    int a = find(parent, x);
    int b = find(parent, y);
    parent[a] = b;
}

int main() {
    int n = 4; // vertices
    int edges = 5;
    int cost[5][3] = {
        {0,1,10},
        {0,2,6},
```

```
    {0,3,5},
    {1,3,15},
    {2,3,4}
  };

  int parent[n];
  for(int i=0;i<n;i++)
    parent[i] = i;

  int mincost = 0;
  printf("Edges in MST:\n");

  for(int i=0;i<edges;i++) {
    int u = cost[i][0];
    int v = cost[i][1];
    int w = cost[i][2];

    if(find(parent, u) != find(parent, v)) {
      printf("%d - %d : %d\n", u, v, w);
      mincost += w;
      union_set(parent, u, v);
    }
  }

  printf("Minimum Cost = %d", mincost);
  return 0;
}
```

✅ SET H

1. Add PRN at Last in Linked List

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

struct node{
  int prn;
  struct node *next;
};

int main() {
  struct node *head=NULL, *temp, *newnode;
```

```
    int n;

    printf("Enter number of initial students: ");
    scanf("%d",&n);

    for(int i=0;i<n;i++) {
        temp = (struct node*)malloc(sizeof(struct node));
        printf("Enter PRN: ");
        scanf("%d",&temp->prn);
        temp->next = head;
        head = temp;
    }

    newnode = (struct node*)malloc(sizeof(struct node));
    printf("Enter new student PRN: ");
    scanf("%d", &newnode->prn);
    newnode->next = NULL;

    temp = head;
    while(temp->next != NULL)
        temp = temp->next;
    temp->next = newnode;

    printf("PRN added at end.");
    return 0;
}
```

2. Stack – Push & Pop

```
#include <stdio.h>
#define SIZE 5

int stack[SIZE], top = -1;

void push(int x) {
    if(top == SIZE-1)
        printf("Stack Overflow\n");
    else
        stack[++top] = x;
}
```

```
void pop() {
    if(top == -1)
        printf("Stack Underflow\n");
    else
        printf("Popped: %d\n", stack[top--]);
}

int main() {
    push(10);
    push(20);
    pop();
    push(30);
    pop();

    return 0;
}
```

✅ SET I

1. Compare three numbers – find MIN & MAX

```
#include <stdio.h>
int main() {
    int a, b, c;
    printf("Enter three numbers: ");
    scanf("%d %d %d", &a, &b, &c);

    int max = a, min = a;

    if(b > max) max = b;
    if(c > max) max = c;

    if(b < min) min = b;
    if(c < min) min = c;

    printf("Maximum = %d\nMinimum = %d\n", max, min);
    return 0;
}
```

2. Matrix Multiplication

```
#include <stdio.h>

int main() {

   int a[10][10], b[10][10], c[10][10];

   int r1, c1, r2, c2;


   printf("Enter rows & cols of Matrix 1: ");

   scanf("%d %d", &r1, &c1);

   printf("Enter rows & cols of Matrix 2: ");

   scanf("%d %d", &r2, &c2);


   if(c1 != r2) {

      printf("Multiplication Not Possible!");

      return 0;

   }


   printf("Enter Matrix 1:\n");

   for(int i=0;i<r1;i++)

      for(int j=0;j<c1;j++)

         scanf("%d",&a[i][j]);


   printf("Enter Matrix 2:\n");

   for(int i=0;i<r2;i++)

      for(int j=0;j<c2;j++)

         scanf("%d",&b[i][j]);


   for(int i=0;i<r1;i++)

      for(int j=0;j<c2;j++) {

         c[i][j] = 0;

         for(int k=0;k<c1;k++)

            c[i][j] += a[i][k] * b[k][j];

      }


   printf("Result Matrix:\n");

   for(int i=0;i<r1;i++) {

      for(int j=0;j<c2;j++)

         printf("%d ", c[i][j]);

      printf("\n");

   }
```

```
    return 0;
}
```

✅ SET J

1. Read & Display Marks

```
#include <stdio.h>
int main() {
    int n;
    printf("Enter number of students: ");
    scanf("%d", &n);

    int marks[n];
    for(int i=0;i<n;i++) {
        printf("Enter marks of student %d: ", i+1);
        scanf("%d", &marks[i]);
    }

    printf("\nMarks of Students:\n");
    for(int i=0;i<n;i++)
        printf("Student %d = %d\n", i+1, marks[i]);

    return 0;
}
```

2. Queue – Enqueue & Dequeue

```
#include <stdio.h>
#define SIZE 5

int queue[SIZE], front = -1, rear = -1;

void enqueue(int x) {
    if(rear == SIZE-1)
        printf("Queue Overflow\n");
    else {
        if(front == -1) front = 0;
        queue[++rear] = x;
    }
}

void dequeue() {
    if(front == -1)
```

```
        printf("Queue Underflow\n");
    else {
        printf("Deleted: %d\n", queue[front]);
        if(front == rear)
            front = rear = -1;
        else
            front++;
    }
}

int main() {
    enqueue(10);
    enqueue(20);
    enqueue(30);

    dequeue();
    dequeue();

    return 0;
}
```

✅ SET K

1. Add Two Matrices

```
#include <stdio.h>
int main() {
    int r, c, a[10][10], b[10][10], s[10][10];

    printf("Enter rows & columns: ");
    scanf("%d %d", &r, &c);

    printf("Enter Matrix 1:\n");
    for(int i=0;i<r;i++)
        for(int j=0;j<c;j++)
            scanf("%d",&a[i][j]);

    printf("Enter Matrix 2:\n");
    for(int i=0;i<r;i++)
        for(int j=0;j<c;j++)
            scanf("%d",&b[i][j]);
```

```
    for(int i=0;i<r;i++)
        for(int j=0;j<c;j++)
            s[i][j] = a[i][j] + b[i][j];

    printf("Sum Matrix:\n");
    for(int i=0;i<r;i++) {
        for(int j=0;j<c;j++)
            printf("%d ", s[i][j]);
        printf("\n");
    }
    return 0;
}
```

2. Fibonacci Series

```
#include <stdio.h>
int main() {
    int n, a = 0, b = 1, c;

    printf("Enter number of terms: ");
    scanf("%d", &n);

    printf("Fibonacci Series: ");
    for(int i=1;i<=n;i++) {
        printf("%d ", a);
        c = a + b;
        a = b;
        b = c;
    }
    return 0;
}
```

✅ SET L

1. Binary Search – CGPA = 6.0

```
#include <stdio.h>
int main() {
    int n;
    printf("Enter number of students: ");
    scanf("%d",&n);

    float cgpa[n];
    printf("Enter CGPAs (sorted):\n");
```

```
    for(int i=0;i<n;i++)
        scanf("%f",&cgpa[i]);

    float key = 6.0;
    int low=0, high=n-1, mid, roll=-1;

    while(low <= high) {
        mid = (low+high)/2;
        if(cgpa[mid] == key) {
            roll = mid+1;
            break;
        }
        else if(cgpa[mid] < key)
            low = mid + 1;
        else
            high = mid - 1;
    }

    if(roll != -1)
        printf("Roll No with CGPA 6.0 = %d", roll);
    else
        printf("CGPA 6.0 not found");

    return 0;
}
```

2. Binary Tree

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

struct node {
    int data;
    struct node *left, *right;
};

struct node* create(int x) {
    struct node* n = (struct node*)malloc(sizeof(struct node));
    n->data = x;
    n->left = n->right = NULL;
    return n;
```

```
}

struct node* insert(struct node* root, int x) {
   if(root == NULL)
      return create(x);

   if(x < root->data)
      root->left = insert(root->left, x);
   else
      root->right = insert(root->right, x);

   return root;
}

void inorder(struct node* root) {
   if(root != NULL) {
      inorder(root->left);
      printf("%d ",root->data);
      inorder(root->right);
   }
}

int main() {
   struct node* root = NULL;

   root = insert(root, 50);
   insert(root, 30);
   insert(root, 20);
   insert(root, 40);
   insert(root, 70);

   printf("Inorder Traversal: ");
   inorder(root);

   return 0;
}
```

✅ SET M

1. Kruskal's Minimum Spanning Tree

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

```
int find(int parent[], int i) {
   while(parent[i] != i)
      i = parent[i];
   return i;
}

void union_set(int parent[], int x, int y) {
   int a = find(parent, x);
   int b = find(parent, y);
   parent[a] = b;
}

int main() {
   int n = 4;
   int edges = 5;

   int cost[5][3] = {
      {0,1,10},
      {0,2,6},
      {0,3,5},
      {1,3,15},
      {2,3,4}
   };

   int parent[n];
   for(int i=0;i<n;i++)
      parent[i] = i;

   int mincost = 0;
   printf("Edges in MST:\n");

   for(int i=0;i<edges;i++) {
      int u = cost[i][0];
      int v = cost[i][1];
      int w = cost[i][2];

      if(find(parent, u) != find(parent, v)) {
         printf("%d - %d : %d\n",u,v,w);
```

```
            mincost += w;
            union_set(parent,u,v);
        }
    }

    printf("Minimum Cost = %d",mincost);
    return 0;
}
```

2. 1D Array – Sum

```
#include <stdio.h>
int main() {
    int n, sum = 0;

    printf("Enter size: ");
    scanf("%d",&n);

    int a[n];
    for(int i=0;i<n;i++) {
        printf("Enter element %d: ",i+1);
        scanf("%d",&a[i]);
        sum += a[i];
    }

    printf("Sum of array = %d", sum);
    return 0;
}
```