SET A

1. Fibonacci Series

```
#include <stdio.h>

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

    printf("Fibonacci Series: %d %d ", a, b);
    for(i = 2; i < n; i++) {
        c = a + b;
        printf("%d ", c);
        a = b;
        b = c;
    }
    return 0;
}
```

2. Read & Display Marks

```
#include <stdio.h>

int main() {
    int n, i, marks[50];
    printf("Enter number of students: ");
    scanf("%d", &n);

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

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

    return 0;
}
```

SET B

1. Min & Max of 3 Numbers

```
#include <stdio.h>

int main() {
    int a, b, c;
    printf("Enter 3 numbers: ");
    scanf("%d %d %d", &a, &b, &c);
    int min = a, max = a;

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

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

    printf("Min = %d\nMax = %d", min, max);
    return 0;
}
```

2. Matrix Multiplication

```
#include <stdio.h>

int main() {
    int a[10][10], b[10][10], m, n, p, q, i, j, k, res[10][10];

    printf("Enter rows & cols of matrix A: ");
    scanf("%d %d", &m, &n);
    printf("Enter rows & cols of matrix B: ");
    scanf("%d %d", &p, &q);

    if(n != p) {
        printf("Multiplication not possible");
        return 0;
    }

    printf("Enter matrix A:\n");
    for(i=0;i<m;i++)
        for(j=0;j<n;j++)
            scanf("%d",&a[i][j]);

    printf("Enter matrix B:\n");
    for(i=0;i<p;i++)
        for(j=0;j<q;j++)
            scanf("%d",&b[i][j]);

    for(i=0;i<m;i++)
        for(j=0;j<q;j++) {
            res[i][j] = 0;
            for(k=0;k<n;k++)
                res[i][j] += a[i][k] * b[k][j];
        }

    printf("Result:\n");
    for(i=0;i<m;i++){
        for(j=0;j<q;j++)
            printf("%d ",res[i][j]);
        printf("\n");
    }
    return 0;
}
```

SET C

1. 1D Array Sum

```
#include <stdio.h>

int main() {
    int n, i, a[50], sum = 0;

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

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

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

2. Add Two Matrices

```
#include <stdio.h>

int main() {
    int a[10][10], b[10][10], c[10][10], m, n, i, j;

    printf("Enter rows & cols: ");
    scanf("%d %d", &m, &n);

    printf("Enter matrix A:\n");
    for(i=0;i<m;i++)
        for(j=0;j<n;j++)
            scanf("%d",&a[i][j]);

    printf("Enter matrix B:\n");
    for(i=0;i<m;i++)
        for(j=0;j<n;j++)
            scanf("%d",&b[i][j]);

    printf("Sum:\n");
    for(i=0;i<m;i++){
        for(j=0;j<n;j++){
            c[i][j] = a[i][j] + b[i][j];
            printf("%d ", c[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

SET D

1. Bubble Sort (Age Sorting)

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

```
int main() {
    int n, a[50], i, j, temp;

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

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

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

    printf("Sorted Age: ");
    for(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, *ptr;
    int n, val, del;

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

    while(n--) {
        temp = (struct node*)malloc(sizeof(struct node));
        scanf("%d", &temp->prn);
        temp->next = head;
        head = temp;
    }

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

    temp = head;
    ptr = NULL;

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

    if(temp == NULL)
```

```
        printf("PRN not found");
    else {
        if(ptr == NULL) head = temp->next;
        else ptr->next = temp->next;
        free(temp);
        printf("Deleted.\n");
    }

    return 0;
}
```

SET E

1. Quick Sort (Age Sorting)

```
#include <stdio.h>

int partition(int a[], int low, int high) {
    int pivot = a[high], i = low-1, j, temp;

    for(j=low;j<high;j++) {
        if(a[j] <= pivot) {
            i++;
            temp=a[i]; a[i]=a[j]; a[j]=temp;
        }
    }
    temp=a[i+1]; a[i+1]=a[high]; a[high]=temp;
    return i+1;
}

void quicksort(int a[], int low, int high) {
    if(low < high) {
        int pi = partition(a, low, high);
        quicksort(a, low, pi-1);
        quicksort(a, pi+1, high);
    }
}

int main() {
    int a[50], n, i;
    scanf("%d",&n);

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

    quicksort(a,0,n-1);

    for(i=0;i<n;i++) printf("%d ",a[i]);
}
```

2. Queue – Enqueue & Dequeue

```
#include <stdio.h>
#define SIZE 50

int main() {
    int q[SIZE], front = -1, rear = -1, choice, val;
    while(1) {
        printf("\n1.Enqueue 2.Dequeue 3.Exit: ");
        scanf("%d", &choice);

        if(choice == 1) {
            printf("Enter value: ");
            scanf("%d", &val);
            if(rear == SIZE-1) printf("Queue Full");
            else {
                if(front == -1) front = 0;
                q[++rear] = val;
            }
        }

        else if(choice == 2) {
            if(front == -1 || front > rear) printf("Queue Empty");
            else printf("Dequeued: %d\n", q[front++]);
        }

        else break;
    }
}
```

Here are **SET F to SET M** programs in clean, simple, exam-ready C code.

SET F

1. Linear Search (Find CGPA = 9.5)

```
#include <stdio.h>

int main() {
    float cgpa[50], key = 9.5;
    int n, i;

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

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

    for(i=0;i<n;i++) {
        if(cgpa[i] == key) {
            printf("Student found at roll no: %d\n", i+1);
            return 0;
        }
    }

    printf("CGPA 9.5 not found");
    return 0;
}
```

2. Binary Tree Implementation

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

```c
#include <stdlib.h>

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

struct node* create() {
    int x;
    printf("Enter data (-1 for no node): ");
    scanf("%d", &x);

    if(x == -1) return NULL;

    struct node* newnode = (struct node*)malloc(sizeof(struct node));
    newnode->data = x;

    printf("Enter left child of %d:\n", x);
    newnode->left = create();

    printf("Enter right child of %d:\n", x);
    newnode->right = create();

    return newnode;
}

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

int main() {
    struct node* root = create();
    printf("Inorder Traversal: ");
    inorder(root);
    return 0;
}
```

SET G

1. Binary Search (Find CGPA = 6.0)

(Array must be sorted)

```c
#include <stdio.h>

int main() {
    float cgpa[50], key = 6.0;
    int n, i, low, high, mid;

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

    printf("Enter sorted CGPA list:\n");
    for(i=0;i<n;i++) scanf("%f",&cgpa[i]);
    low = 0;
    high = n - 1;

    while(low <= high) {
        mid = (low + high) / 2;
        if(cgpa[mid] == key) {
            printf("Student found at roll: %d", mid+1);
            return 0;
        }
        else if(cgpa[mid] < key)
            low = mid + 1;
        else
            high = mid - 1;
    }

    printf("CGPA 6.0 not found");
    return 0;
}
```

2. Minimum Spanning Tree – Kruskal

```
#include <stdio.h>

int parent[20];

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

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

int main() {
    int n, cost[20][20], i, j, edges = 0, min, a, b, u, v;

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

    printf("Enter cost adjacency matrix:\n");
    for(i=0;i<n;i++)
        for(j=0;j<n;j++)
            scanf("%d",&cost[i][j]);

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

    printf("Edges in MST:\n");

    while(edges < n-1) {
        min = 999;
        for(i=0;i<n;i++)
            for(j=0;j<n;j++)
                if(cost[i][j] < min && cost[i][j] != 0) {
                    min = cost[i][j];
                    a = u = i;
                    b = v = j;
                }

        u = find(u);
        v = find(v);

        if(u != v) {
            printf("%d - %d = %d\n", a, b, min);
            union_set(u, v);
            edges++;
        }

        cost[a][b] = cost[b][a] = 999;
    }

    return 0;
}
```

SET H

1. Singly Linked List – Insert at Last

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

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

int main() {
    struct node *head=NULL, *temp, *ptr;
    int n, val;

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

    while(n--) {
        temp = (struct node*)malloc(sizeof(struct node));
        scanf("%d",&temp->prn);
        temp->next = head;
        head = temp;
    }

    printf("Enter new PRN to add at last: ");
    scanf("%d",&val);

    temp = (struct node*)malloc(sizeof(struct node));
    temp->prn = val;
    temp->next = NULL;

    if(head == NULL) head = temp;
    else {
        ptr = head;
        while(ptr->next != NULL) ptr = ptr->next;
        ptr->next = temp;
    }

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

2. Stack – Push, Pop

```
#include <stdio.h>
#define SIZE 50

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() {
    int ch, val;

    while(1) {
        printf("\n1.Push 2.Pop 3.Exit: ");
        scanf("%d",&ch);

        if(ch==1) {
            scanf("%d",&val);
            push(val);
        }
        else if(ch==2) pop();
        else break;
    }
}
```

SET I

(Same as SET B – repeated)

1. Min & Max of 3 Numbers → already given

2. Matrix Multiplication → already given

(Use programs from SET B)

SET J

1. Read & Display Marks

(Already in SET A #2)

2. Queue – Enqueue/Dequeue

(Already in SET E #2)

SET K

1. Add Two Matrices

(Already in SET C #2)

2. Fibonacci Series

(Already in SET A #1)

SET L

1. Binary Search for CGPA = 6.0

(Already in SET G #1)

2. Binary Tree Implementation

(Already in SET F #2)

SET M

1. Minimum Spanning Tree – Kruskal

(Already in SET G #2)

2. 1D Array Sum

(Already in SET C #1)

If you want, I can also give **all codes in one single PDF** or **explain logic in short bullet points** for exam revision.