1) WRITE C PATTERN USING C LANGUAGE

1. Right-angle triangle with a number

```
#include <stdio.h>
int main()
{
    int i, j;
    int rows = 5;
    printf("Right-angle triangle with a number\n\n");
    for(i = 1; i <= rows; i++)
    {
        for(j = 1; j <= i; j++)
        {
            printf("%d", j);
        }
        printf("\n");
    }
    return 0;
}
```

2. Diamond shape with a number

```
#include <stdio.h>
int main()
{
    int i, j, n;
    printf("Enter the maximum number: ");
    scanf("%d", &n);
    for(i = 1; i <= n; i += 2)
    {
        for(j = 1; j <= (n - i) / 2; j++)
            printf(" ");
        for(j = 1; j <= i; j++)
            printf("%d ", j);
        printf("\n");
    }
    for(i = n - 2; i >= 1; i -= 2)
    {
        for(j = 1; j <= (n - i) / 2; j++)
            printf(" ");
        for(j = 1; j <= i; j++)
            printf("%d ", j);
        printf("\n");
    }
    return 0;
}
```

3. Pyramid with an asterisk

```
#include <stdio.h>
int main()
{
    int i, j, space, rows;
    printf("Enter number of rows: ");
    scanf("%d", &rows);
    for(i = 1; i <= rows; i++)
    {
        for(space = 1; space <= rows - i; space++)
        {
            printf(" ");
        }
```

```
        for(j = 1; j <= i; j++)
        {
            printf("* ");
        }
        printf("\n");
    }
    return 0;
}
```

4. Pyramid using the alphabets

```
#include <stdio.h>
int main()
{
    int i, j, space, rows;
    printf("Enter number of rows: ");
    scanf("%d", &rows);
    for(i = 1; i <= rows; i++)
    {
        for(space = 1; space <= rows - i; space++)
        {
            printf(" ");
        }
        for(j = 0; j < i; j++)
        {
            printf("%c ", 'A' + j);
        }
        for(j = i - 2; j >= 0; j--)
        {
            printf("%c ", 'A' + j);
        }
        printf("\n");
    }
    return 0;
}
```

---

2) STUDENT DATABASE MANAGEMENT

```
#include <stdio.h>
#include <string.h>
#define MAX_STUDENTS 10
#define NUM_SUBJECTS 3

struct Student {
    int roll_no;
    char name[50];
    char program[30];
    char course[30];
    int subject_marks[NUM_SUBJECTS];
    int total_marks;
    float average_marks;
};

int main()
{
    struct Student students[MAX_STUDENTS];
    int num_students = 0;
    int i, j;

    printf("Enter the number of students (max %d): ", MAX_STUDENTS);
    scanf("%d", &num_students);

    if (num_students > MAX_STUDENTS || num_students <= 0)
```

```
    {
        printf("Invalid number of students.\n");
        return 1;
    }

    getchar();

    for (i = 0; i < num_students; i++)
    {
        printf("\nEnter details for Student %d:\n", i + 1);

        printf("Roll No: ");
        scanf("%d", &students[i].roll_no);
        getchar();

        printf("Name: ");
        fgets(students[i].name, sizeof(students[i].name), stdin);
        students[i].name[strcspn(students[i].name, "\n")] = '\0';

        printf("Program: ");
        fgets(students[i].program, sizeof(students[i].program), stdin);
        students[i].program[strcspn(students[i].program, "\n")] = '\0';

        printf("Course: ");
        fgets(students[i].course, sizeof(students[i].course), stdin);
        students[i].course[strcspn(students[i].course, "\n")] = '\0';

        students[i].total_marks = 0;

        printf("Enter marks for %d subjects:\n", NUM_SUBJECTS);
        for (j = 0; j < NUM_SUBJECTS; j++)
        {
            printf("Subject %d marks: ", j + 1);
            scanf("%d", &students[i].subject_marks[j]);
            students[i].total_marks += students[i].subject_marks[j];
        }

        students[i].average_marks = (float)students[i].total_marks / NUM_SUBJECTS;
        getchar();
    }

    printf("\n--- Student Database ---\n");
    printf("Roll
No.\tName\t\tProgram\t\tCourse\t\tSub1\tSub2\tSub3\tTotal\tAverage\n");
    printf("--------\t----\t\t-------\t\t------\t\t----\t----\t----\t-----\t-------
\n");

    for (i = 0; i < num_students; i++)
    {
        printf("%-8d\t%-15s\t%-15s\t%-15s",
               students[i].roll_no,
               students[i].name,
               students[i].program,
               students[i].course);

        for (j = 0; j < NUM_SUBJECTS; j++)
        {
            printf("\t%d", students[i].subject_marks[j]);
        }

        printf("\t%d\t%.2f\n", students[i].total_marks, students[i].average_marks);
    }

    return 0;
}
```

3) SEARCHING TECHNIQUES

1. Linear Search

```
#include <stdio.h>

int main()
{
    int i, n, key, a[100];
    int flag = 0;

    printf("\n************* Program for Linear Search *************");
    printf("\n\nNumber of element in an array n=:");
    scanf("%d", &n);

    printf("\nElement in an array: ");
    for(i = 0; i < n; i++)
    {
        scanf("%d", &a[i]);
    }

    printf("\n\nkey element to search in an array n=:");
    scanf("%d", &key);

    printf("\nOriginal array is: ");
    for(i = 0; i < n; i++)
    {
        printf("\na[%d]= %d", i, a[i]);
    }

    for(i = 0; i < n; i++)
    {
        if(a[i] == key)
        {
            flag = 1;
            break;
        }
    }

    if(flag == 1)
        printf("\n\nkey=%d is present at %d location", key, i);
    else
        printf("\n\nKey=%d is not present in given array", key);

    return 0;
}
```

2. Binary Search

```
#include <stdio.h>

int main()
{
    int i, n, key, a[100];
    int low, high, m, flag = 0;

    printf("\n************* Program for Binary Search *************");
    printf("\n\nNumber of element in an array n=:");
    scanf("%d", &n);

    printf("\nElement in an array: ");
    for(i = 0; i < n; i++)
    {
```

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

    printf("\n\nkey element to search in an array n=:");
    scanf("%d", &key);

    printf("\nOriginal array is: ");
    for(i = 0; i < n; i++)
    {
        printf("\na[%d]= %d", i, a[i]);
    }

    low = 0;
    high = n - 1;

    while(low <= high)
    {
        m = (low + high) / 2;

        if(a[m] == key)
        {
            flag = 1;
            break;
        }
        else if(key < a[m])
            high = m - 1;
        else
            low = m + 1;
    }

    if(flag == 1)
        printf("\n\nkey=%d is present at a[%d] location", key, m);
    else
        printf("\nKey=%d is not present in given array", key);

    return 0;
}
```

---

4) SORTING ALGORITHMS

1. Bubble Sort

```
#include <stdio.h>

int main()
{
    int i, j, n, temp, a[100];

    printf("\n************* program for Bubble Sort *************");
    printf("\n\nNumber of element in an array n=:");
    scanf("%d", &n);

    printf("\nElement in an array: ");
    for(i = 0; i < n; i++)
    {
        scanf("%d", &a[i]);
    }

    printf("\nOriginal array is: ");
    for(i = 0; i < n; i++)
    {
        printf("\n%d", a[i]);
    }
```

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

    printf("\n\nBubble sorted array is:");
    for(i = 0; i < n; i++)
    {
        printf("\n%d", a[i]);
    }

    return 0;
}
```

2. Selection Sort

```
#include <stdio.h>

int main()
{
    int i, j, n, temp, min, a[100];

    printf("\n************* Program for Selection Sort *************");
    printf("\n\nNumber of element in an array n=:");
    scanf("%d", &n);

    printf("\nElement in an array: ");
    for(i = 0; i < n; i++)
    {
        scanf("%d", &a[i]);
    }

    printf("\nOriginal array is:");
    for(i = 0; i < n; i++)
    {
        printf("\n%d", a[i]);
    }

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

        if(min != i)
        {
            temp = a[min];
            a[min] = a[i];
            a[i] = temp;
        }
    }

    printf("\n\nSelection sorted array is:");
    for(i = 0; i < n; i++)
```

```
    {
        printf("\n%d", a[i]);
    }

    return 0;
}
```

3. Insertion Sort

```
#include <stdio.h>

int main()
{
    int i, j, n, temp, a[100];

    printf("\n************* Program for Insertion Sort *************");
    printf("\n\nNumber of element in an array n=:");
    scanf("%d", &n);

    printf("\nElement in an array: ");
    for(i = 0; i < n; i++)
    {
        scanf("%d", &a[i]);
    }

    printf("\nOriginal array is: ");
    for(i = 0; i < n; i++)
    {
        printf("\n%d", a[i]);
    }

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

        while(j >= 0 && a[j] > temp)
        {
            a[j + 1] = a[j];
            j--;
        }
        a[j + 1] = temp;
    }

    printf("\n\nInsertion sorted array is:");
    for(i = 0; i < n; i++)
    {
        printf("\n%d", a[i]);
    }

    return 0;
}
```

---

5) STACK AND QUEUE USING ARRAY

1. Stack Using Array

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 10

int disp();
```

```
int pop();
int push();
int isEmpty();
int isFull();

int stack[SIZE];
int top = -1;

int main()
{
    int ch;

    while(1)
    {
        printf("\n****** Select Stack Operation **********");
        printf("\n1.Push \n2.Pop \n3.Display \n4.Exit ");
        printf("\nEnter your Choice:=");
        scanf("%d", &ch);

        switch(ch)
        {
            case 1: push();
                    break;
            case 2: pop();
                    break;
            case 3: disp();
                    break;
            case 4: exit(0);
                    break;
        }
    }
    return 0;
}

int isEmpty()
{
    if(top == -1)
        return 1;
    else
        return 0;
}

int isFull()
{
    if(top == SIZE - 1)
        return 1;
    else
        return 0;
}

int push()
{
    int item;

    if (isFull())
    {
        printf("Stack Full! Cannot push");
    }
    else
    {
        printf("Enter element in stack= ");
        scanf("%d", &item);
        top++;
        stack[top] = item;
    }
}
```

```
int pop()
{
    int popped_item;

    if(isEmpty())
    {
        printf("Stack Underflow! Cannot pop from empty stack.\n");
    }
    else
    {
        popped_item = stack[top];
        top--;
        printf("Popped element is %d", popped_item);
    }
}

int disp()
{
    int i;

    if(isEmpty())
    {
        printf("Stack is empty.\n");
    }
    else
    {
        printf("Elements in stack: ");
        for(i = top; i >= 0; i--)
        {
            printf("%d ", stack[i]);
        }
        printf("\n");
    }
}
```

2. Queue Using Array

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

int disp();
int dequeue();
int enqueue();
int isEmpty();
int isFull();

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

int main()
{
    int ch;

    while(1)
    {
        printf("\n****** Select Queue Operation **********");
        printf("\n1.Enqueue add element \n2.Dequeue Delete Element \n3.Display \n4.Exit
");
        printf("\nEnter your Choice:=");
        scanf("%d", &ch);

        switch(ch)
        {
            case 1: enqueue();
                    break;
```

```
            case 2: dequeue();
                    break;
            case 3: disp();
                    break;
            case 4: exit(0);
                    break;
        }
    }
    return 0;
}

int isEmpty()
{
    if (front == -1 || front > rear)
        return 1;
    else
        return 0;
}

int isFull()
{
    if (rear == SIZE - 1)
        return 1;
    else
        return 0;
}

int enqueue()
{
    int item;

    if (isFull())
    {
        printf("Queue is Full! Insertion not possible.\n");
    }
    else
    {
        if (front == -1)
        {
            front = 0;
        }
        printf("Enter element in Queue= ");
        scanf("%d", &item);
        rear++;
        queue[rear] = item;
        printf("%d enqueued to queue.\n", item);
    }
}

int dequeue()
{
    int deleted_value;

    if (isEmpty())
    {
        printf("Queue is Empty! Deletion not possible.\n");
    }
    else
    {
        deleted_value = queue[front];
        front++;

        if (front > rear)
        {
            front = -1;
            rear = -1;
        }
```

```
        printf("%d dequeued from queue.\n", deleted_value);
    }
}

int disp()
{
    int i;

    if (front == -1)
        printf("Queue is Empty!\n");
    else
    {
        printf("Queue elements: ");
        for (i = front; i <= rear; i++)
        {
            printf("%d ", queue[i]);
        }
        printf("\n");
    }
}
```

---

6) CONVERT DECIMAL NUMBER TO A BINARY NUMBER

```
#include <stdio.h>

int main()
{
    int i, decimal;

    printf("Enter a decimal number: ");
    scanf("%d", &decimal);

    int binary[32];
    int quotient, top = -1;

    for(quotient = decimal; quotient > 0; quotient /= 2)
    {
        top++;
        binary[top] = quotient % 2;
    }

    printf("Result Binary equivalent: ");
    for(i = top; i >= 0; i--)
    {
        printf("%d", binary[i]);
    }
    printf("\n");

    return 0;
}
```

---

7) SINGLY LINKED LIST

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

typedef struct node {
    int data;
    struct node *next;
```

```
} node;

node *create();
void disp(node*);
int search(node*);
node *insert(node*);
node *del(node*);

void main()
{
    node *first = NULL;
    int ch;

    do
    {
        printf("\n****** Select SLL Operation **********");
        printf("\n1.Create \n2.Display \n3.Search \n4.Insert \n5.Delete \n6.Exit ");
        printf("\nEnter your Choice:=");
        scanf("%d", &ch);

        switch(ch)
        {
            case 1: first = create();
                    break;
            case 2: disp(first);
                    break;
            case 3: search(first);
                    break;
            case 4: first = insert(first);
                    break;
            case 5: first = del(first);
                    break;
        }
    } while(ch != 6);
}

node *create()
{
    node *first, *temp, *ptr;
    int i, x, n;

    first = NULL;

    printf("\nNumber of items:");
    scanf("%d", &n);

    for(i = 1; i <= n; i++)
    {
        printf("\nenter data=");
        scanf("%d", &x);

        ptr = (struct node*)malloc(sizeof(struct node));
        ptr->data = x;
        ptr->next = NULL;

        if(first == NULL)
        {
            first = ptr;
        }
        else
        {
            temp->next = ptr;
        }
        temp = ptr;
    }
    return(first);
}
```

```
void disp(node *temp)
{
    while(temp != NULL)
    {
        printf(" %d->", temp->data);
        temp = temp->next;
    }
    printf("NULL");
}

int search(node *temp)
{
    int x, index = 0;

    printf("\nEnter number to search=");
    scanf("%d", &x);

    while(temp != NULL && temp->data != x)
    {
        temp = temp->next;
        index++;
    }

    if(temp == NULL)
        printf("\n%d is not found", x);
    else
        printf("\n%d is found at location %d", x, index);
}

node *insert(node *first)
{
    node *temp, *ptr;
    int x, pos, i = 1;

    temp = first;

    printf("\nEnter position to insert=");
    scanf("%d", &pos);
    printf("\nEnter data to insert=");
    scanf("%d", &x);

    while(i < pos && temp != NULL)
    {
        temp = temp->next;
        i++;
    }

    if (temp != NULL || pos == 0)
    {
        ptr = (node*)malloc(sizeof(node));
        ptr->data = x;
        ptr->next = NULL;

        if(pos == 0)
        {
            ptr->next = first;
            first = ptr;
        }
        else
        {
            ptr->next = temp->next;
            temp->next = ptr;
        }
    }
    else
        printf("\nInvalid Position");
```

```
    return(first);
}

node *del(node *first)
{
    node *temp, *prev;
    int pos, i = 0;

    temp = first;

    printf("\nEnter node number to delete=");
    scanf("%d", &pos);

    while(i < pos && temp != NULL)
    {
        prev = temp;
        temp = temp->next;
        i++;
    }

    if (temp != NULL)
    {
        if(pos == 0)
        {
            first = first->next;
            free(temp);
        }
        else
        {
            prev->next = temp->next;
            free(temp);
        }
    }
    else
        printf("\nInvalid Position");

    return(first);
}
```