C Programs

//1&2.Write a program in C to display the following patterns like Right-angle triangle Diamond shape Pyramid with Pyramid using with a number: with numbers; an asterisk: the alphabet A

```
#include <stdio.h>

int main() {
   int i, j, k, n;
   n = 4;
 printf("Right-angle triangle with numbers:\n");
   for(i = 1; i <= n; i++) {
      for(j = 1; j <= i; j++) {
         printf("%d", j);
      }
      printf("\n");
   }
 printf("\nDiamond shape with numbers:\n");
   for(i = 1; i <= n; i++) {
      for(j = i; j < n; j++)
         printf(" ");
      for(k = 1; k <= i; k++)
         printf("%d ", i);
      printf("\n");
   }
   for(i = n - 1; i >= 1; i--) {
      for(j = n; j > i; j--)
```

```
        printf(" ");
      for(k = 1; k <= i; k++)
        printf("%d ", i);
      printf("\n");
   }
    printf("Pyramid with asterisks:\n");
   for(i = 1; i <= n; i++) {
      for(j = i; j < n; j++)
        printf(" ");
      for(k = 1; k <= (2 * i - 1); k++)
        printf("* ");
      printf("\n");
   }
printf("\nPyramid with alphabets:\n");
   for(i = 1; i <= n; i++) {
      for(j = i; j < n; j++)
        printf(" ");
      char ch = 'A';
      for(k = 1; k <= i; k++)
        printf("%c ", ch++);
      ch -= 2;
      for(k = 1; k < i; k++)
        printf("%c ", ch--);
      printf("\n");
   }
 return 0;
```

```
}

//3.Student Result Management System using Array of Structures in C.
#include <stdio.h>
struct Student {
   int rollno;
   char name[50];
   float marks[3];
   float total;
   float percentage;
};
int main() {
   struct Student s[50];
   int n, i, j;
     printf("Enter number of students: ");
   scanf("%d", &n);
    for(i = 0; i < n; i++) {
   printf("\nEnter details for student %d:\n", i +1);
      printf("Roll Number: ");
      scanf("%d", &s[i].rollno);
      printf("Name: ");
      scanf("%s", s[i].name);
       s[i].total = 0;
      for(j = 0; j < 3; j++) {
         printf("Enter marks in subject %d: ", j + 1);
         scanf("%f", &s[i].marks[j]);
```

```
            s[i].total += s[i].marks[j];
        }
        s[i].percentage = s[i].total / 3.0;
    }
   printf("\n---------------- STUDENT RESULT ----------------\n");
printf("RollNo\tName\tSub1\tSub2\tSub3\tTotal\tPercent\tGrade\n");
    printf("-------------------------------------------------\n");
     for(i = 0; i < n; i++) {
        char grade;
        if(s[i].percentage >= 75)
            grade = 'A';
        else if(s[i].percentage >= 60)
            grade = 'B';
        else if(s[i].percentage >= 50)
            grade = 'C';
        else if(s[i].percentage >= 40)
            grade = 'D';
        else
            grade = 'F';
          printf("%d\t%s\t", s[i].rollno, s[i].name);
        for(j = 0; j < 3; j++)
            printf("%.1f\t", s[i].marks[j]);
        printf("%.1f\t%.1f\t%c\n", s[i].total, s[i].percentage, grade);
    }
     return 0;
}
```

//4.Design and implement a program to search into interger array using Linerar / Binary search

```
#include <stdio.h>
int linearSearch(int arr[], int n, int key) {
    for (int i = 0; i < n; i++) {
    if (arr[i] == key)
   return i;
   }
 return -1;
}
int binarySearch(int arr[], int n, int key) {
  int left = 0, right = n - 1, mid;
while (left <= right) {
     mid = (left + right) / 2;
    if (arr[mid] == key)
    return mid;
    else if (arr[mid] < key)
       left = mid + 1;
     else
   right = mid - 1;
    }
return -1;
}
int main() {
    int n, key, choice;
```

```
 printf("Enter number of elements: ");
   scanf("%d", &n);
   int arr[n];
   printf("Enter %d integers: ", n);
   for (int i = 0; i < n; i++) {
      scanf("%d", &arr[i]);
   }
printf("Enter element to search: ");
   scanf("%d", &key);
printf("\nChoose search method:\n");
   printf("1. Linear Search\n");
   printf("2. Binary Search (Array must be sorted)\n");
   printf("Enter choice (1 or 2): ");
   scanf("%d", &choice);
 int result;
   if (choice == 1) {
      result = linearSearch(arr, n, key);
      if (result != -1)
         printf("Element found at position %d (index %d)\n", result + 1, result);
      else
         printf("Element not found in array.\n");
   } else if (choice == 2) {
      result = binarySearch(arr, n, key);
      if (result != -1)
         printf("Element found at position %d (index %d)\n", result + 1, result);
      else
```

```
        printf("Element not found in array.\n");
    } else {
        printf("Invalid choice!\n");
    }
 return 0;
}
```

//5.Design and implement a program to sort given array using Bubble sort using user define function

```
#include <stdio.h>
void bubbleSort(int arr[], int n) {
    int i, j, temp;
    for (i = 0; i < n - 1; i++) {
        for (j = 0; j < n - i - 1; j++) {
        if (arr[j] > arr[j + 1]) {
       temp = arr[j];
       arr[j] = arr[j + 1];
        arr[j + 1] = temp;
        }
        }
        }
}
void displayArray(int arr[], int n) {
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
```

```
    }
    printf("\n");
}
int main() {
    int n;
      printf("Enter number of elements: ");
    scanf("%d", &n);
   int arr[n];
    printf("Enter %d integers: ", n);
    for (int i = 0; i < n; i++) {
    scanf("%d", &arr[i]);
    }
   printf("\nOriginal array: ");
    displayArray(arr, n);
    bubbleSort(arr, n);
    printf("Sorted array in ascending order: ");
    displayArray(arr, n);
 return 0;
}

//.6Implement stack using linkedlist.

#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
```

```
    struct Node *next;
};
struct Node *top = NULL;
void push(int val) {
    struct Node *newNode = malloc(sizeof(struct Node));
    newNode->data = val;
    newNode->next = top;
    top = newNode;
    printf("%d pushed\n", val);
}
void pop() {
    if (top == NULL) printf("Stack Underflow\n");
    else {
        struct Node *temp = top;
        printf("%d popped\n", top->data);
        top = top->next;
        free(temp);
    }
}
void display() {
    struct Node *temp = top;
    if (!temp) printf("Stack Empty\n");
    else {
        printf("Stack: ");
        while (temp) {
            printf("%d ", temp->data);
```

```
        temp = temp->next;
      }
      printf("\n");
   }
}

int main() {
   int ch, val;
   while (1) {
      printf("\n1.Push 2.Pop 3.Display 4.Exit\nEnter choice: ");
      scanf("%d", &ch);
      switch (ch) {
         case 1: printf("Enter value: "); scanf("%d", &val); push(val); break;
         case 2: pop(); break;
         case 3: display(); break;
         case 4: exit(0);
         default: printf("Invalid choice\n");
      }
   }
}
```

//7.Design and implement a program to sort given array using Insertion sort using user define function

```
#include <stdio.h>
void insertionSort(int arr[], int n) {
```

```
    int i, key, j;
    for (i = 1; i < n; i++) {
        key = arr[i];
        j = i - 1;
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = key;
    }
}
void display(int arr[], int n) {
    for (int i = 0; i < n; i++)
        printf("%d ", arr[i]);
    printf("\n");
}
int main() {
    int n;
    printf("Enter number of elements: ");
    scanf("%d", &n);
   int arr[n];
    printf("Enter %d integers: ", n);
    for (int i = 0; i < n; i++)
        scanf("%d", &arr[i]);
  printf("\nOriginal array: ");
    display(arr, n);
```

```
    insertionSort(arr, n);
   printf("Sorted array in ascending order: ");
    display(arr, n);
  return 0;
}

//8.Implement Stack using arrays.
#include <stdio.h>
#define SIZE 5
int stack[SIZE], top = -1;
void push(int val) {
    if (top == SIZE - 1) printf("Stack Overflow\n");
    else stack[++top] = val, printf("%d pushed\n", val);
}
void pop() {
    if (top == -1) printf("Stack Underflow\n");
    else printf("%d popped\n", stack[top--]);
}
void display() {
    if (top == -1) printf("Stack Empty\n");
    else {
        printf("Stack: ");
        for (int i = top; i >= 0; i--) printf("%d ", stack[i]);
        printf("\n");
    }
}
```

```
int main() {
   int ch, val;
   while (1) {
      printf("\n1.Push 2.Pop 3.Display 4.Exit\nEnter choice: ");
      scanf("%d", &ch);
      switch (ch) {
         case 1: printf("Enter value: "); scanf("%d", &val); push(val); break;
         case 2: pop(); break;
         case 3: display(); break;
         case 4: return 0;
         default: printf("Invalid choice\n");
      }
   }
}

//.9Implement Queue using arrays.

#include <stdio.h>
#define SIZE 5
int queue[SIZE], front = -1, rear = -1;
void enqueue(int val) {
   if (rear == SIZE - 1) printf("Queue Overflow\n");
   else {
      if (front == -1) front = 0;
      queue[++rear] = val;
      printf("%d enqueued\n", val);
```

```
    }
}
void dequeue() {
    if (front == -1 || front > rear) printf("Queue Underflow\n");
    else printf("%d dequeued\n", queue[front++]);
}
void display() {
    if (front == -1 || front > rear) printf("Queue Empty\n");
    else {
        printf("Queue: ");
        for (int i = front; i <= rear; i++) printf("%d ", queue[i]);
        printf("\n");
    }
}
int main() {
    int ch, val;
    while (1) {
        printf("\n1.Enqueue 2.Dequeue 3.Display 4.Exit\nEnter choice: ");
        scanf("%d", &ch);
        switch (ch) {
            case 1: printf("Enter value: "); scanf("%d", &val); enqueue(val); break;
            case 2: dequeue(); break;
            case 3: display(); break;
            case 4: return 0;
            default: printf("Invalid choice\n");
        }
```

```
    }
}
```

//10.Create a singly linked list with options:a.Insert (at front, at end.)b.Delete (at front, at end)C.Display

```
#include <stdio.h>
#include <stdlib.h>
struct Node { int data; struct Node *next; };
struct Node *head = NULL;
void insertFront(int v){
    struct Node *n = malloc(sizeof(struct Node));
    n->data = v; n->next = head; head = n;
    printf("%d inserted at front\n", v);
}
void insertEnd(int v){
    struct Node *n = malloc(sizeof(struct Node)), *t = head;
    n->data = v; n->next = NULL;
    if(!head) head = n;
    else { while(t->next) t = t->next; t->next = n; }
    printf("%d inserted at end\n", v);
}
void deleteFront(){
    if(!head) printf("List Empty\n");
    else { struct Node *t = head; head = head->next; printf("%d deleted from front\n", t->data); free(t); }
```

```
}
void deleteEnd(){
   if(!head) printf("List Empty\n");
   else if(!head->next){ printf("%d deleted from end\n", head->data); free(head); head=NULL; }
   else { struct Node *t=head,*p; while(t->next){ p=t; t=t->next; } p->next=NULL; printf("%d deleted from end\n", t->data); free(t); }
}
void display(){
   struct Node *t=head;
   if(!t) printf("List Empty\n");
   else { printf("List: "); while(t){ printf("%d ",t->data); t=t->next; } printf("\n"); }
}
int main(){
   int ch,v;
   while(1){
      printf("\n1.InsFront 2.InsEnd 3.DelFront 4.DelEnd 5.Display 6.Exit\nEnter choice: ");
      scanf("%d",&ch);
      switch(ch){
         case 1: printf("Enter value: "); scanf("%d",&v); insertFront(v); break;
         case 2: printf("Enter value: "); scanf("%d",&v); insertEnd(v); break;
         case 3: deleteFront(); break;
         case 4: deleteEnd(); break;
         case 5: display(); break;
         case 6: return 0;
         default: printf("Invalid choice\n");
      }
```

```
    }
}

//11.Implement queue using linkedlist.
#include <stdio.h>
#include <stdlib.h>
struct Node { int data; struct Node *next; };
struct Node *front = NULL, *rear = NULL;
void enqueue(int v){
    struct Node *n = malloc(sizeof(struct Node));
    n->data = v; n->next = NULL;
    if(rear == NULL) front = rear = n;
    else { rear->next = n; rear = n; }
    printf("%d enqueued\n", v);
}
void dequeue(){
    if(front == NULL) printf("Queue Underflow\n");
    else {
        struct Node *t = front;
        printf("%d dequeued\n", t->data);
        front = front->next;
        if(front == NULL) rear = NULL;
        free(t);
    }
}
void display(){
```

```
    if(front == NULL) printf("Queue Empty\n");
    else {
        struct Node *t = front;
        printf("Queue: ");
        while(t){ printf("%d ", t->data); t = t->next; }
        printf("\n");
    }
}
int main(){
    int ch, v;
    while(1){
        printf("\n1.Enqueue 2.Dequeue 3.Display 4.Exit\nEnter choice: ");
        scanf("%d",&ch);
        switch(ch){
            case 1: printf("Enter value: "); scanf("%d",&v); enqueue(v); break;
            case 2: dequeue(); break;
            case 3: display(); break;
            case 4: return 0;
            default: printf("Invalid choice\n");
        }
    }
}

//12.Implement Binary search tree .

#include <stdio.h>
```

```c
#include <stdlib.h>

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

struct Node* createNode(int val) {
   struct Node* newNode = malloc(sizeof(struct Node));
   newNode->data = val;
   newNode->left = newNode->right = NULL;
   return newNode;
}

struct Node* insert(struct Node* root, int val) {
   if (root == NULL) return createNode(val);
   if (val < root->data)
      root->left = insert(root->left, val);
   else if (val > root->data)
      root->right = insert(root->right, val);
   return root;
}

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

```
void preorder(struct Node* root) {
   if (root) { printf("%d ", root->data); preorder(root->left); preorder(root->right); }
}

void postorder(struct Node* root) {
   if (root) { postorder(root->left); postorder(root->right); printf("%d ", root->data); }
}

int main() {
   struct Node* root = NULL;
   int ch, val;
   while (1) {
      printf("\n1.Insert 2.Inorder 3.Preorder 4.Postorder 5.Exit\nEnter choice: ");
      scanf("%d", &ch);
      switch (ch) {
         case 1: printf("Enter value: "); scanf("%d", &val); root = insert(root, val); break;
         case 2: printf("Inorder: "); inorder(root); printf("\n"); break;
         case 3: printf("Preorder: "); preorder(root); printf("\n"); break;
         case 4: printf("Postorder: "); postorder(root); printf("\n"); break;
         case 5: return 0;
         default: printf("Invalid choice\n");
      }
   }
}
```