PROGRAM 1 ELSE IF GRADES

```
#include <stdio.h>

int main() {

int marks;

printf("Enter your marks (0-100): ");

scanf("%d", &marks);

if (marks >= 90) {

printf("Grade: A\n");

}

else if (marks >= 80) {

printf("Grade: B\n");

}

else if (marks >= 70) {

printf("Grade: C\n");

}

else if (marks >= 60) {

printf("Grade: D\n");

}

else {

printf("Grade: F\n");

}

return 0;

}
```

PROGRAM 2 NESTING MULTIPLICATION

```
#include <stdio.h>

int main() {

int i,j,n;

clrscr();

printf("Enter the number:");

scanf("%d",&n);

for (i = 1; i <= n; i++) {

for (j = 1; j <= 10; j++) {

printf("%d\t", i * j);

}

printf("\n");

}

return 0;

}
```

PROGRAM 3 MATRICES

```
#include <stdio.h>

#include <conio.h>

#include <math.h>

int main(){

  int a[10][10], b[10][10], result[10][10], k, i, j, n, c;

  printf("\n Enter the size of the matrix:");

  scanf("%d", &n);

  printf("\n Enter first matrix:");

  for(i = 1; i <= n; i++){

        for(j = 1; j <= n; j++){

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

    }

  }

  printf("\n Enter second matrix:");

  for(i = 1; i <= n; i++){

    for(j = 1; j <= n; j++){

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

    }

  }

  printf("\n matrix 1:");

  for(i = 1; i <= n; i++){

    for(j = 1; j <= n; j++){

      printf("\n %d\t", a[i][j]);

    }

  }
```

```
printf("\n matrix 2:");
for(i = 1; i <= n; i++){
  for(j = 1; j <= n; j++){
    printf("\n %d\t", b[i][j]);
  }
}
printf("\n 1. Add \n 2. Subtract \n 3. Multiplication \n 4. Division");
printf("\n Enter your choice:");
scanf("%d", &c);
switch(c){
  case 1:
    printf("\n 1.Sum");
    for(i = 1; i <= n; i++){
      for(j = 1; j <= n; j++){
        printf("\n %d\t", a[i][j] + b[i][j]);
      }
    }
    break;
  case 2:
    printf("\n 2. Substract");
    for(i = 1; i <= n; i++){
      for(j = 1; j <= n; j++){
        printf("\n %d\t", a[i][j] - b[i][j]);
      }
    }
    break;
    case 3:
        printf("\n 3. Multiplication");
        for(i = 1; i <= n; i++){
            for(j = 1; j <= n; j++){
                result[i][j] = 0;
                for(k = 1; k <= n; k++){
                            result[i][j] = a[i][k]*b[k][j];
                }
            }
        }
        for(i = 1; i <= n; i++){
            for(j = 1; j <= n; j++){
                printf("\n %d", result[i][j]);
            }
        }
        break;
    case 4:
        printf("\n 4. Division");
        for(i = 1; i <= n; i++){
            for(j = 1; j <= n; j++){
                printf("\n %d\t", a[i][j] / b[i][j]);
            }
        }
        break;
}
return 0;
```

```
}
```

PROGRAM 4 TOWER OF HANOI

```
#include <stdio.h>
void towerOfHanoi(int n, char fromRod, char toRod, char auxRod) {
if (n == 1) {
printf("Move disk 1 from %c to %c\n", fromRod, toRod);
return;
}
towerOfHanoi(n - 1, fromRod, auxRod, toRod);
printf("Move disk %d from %c to %c\n", n, fromRod, toRod);
towerOfHanoi(n - 1, auxRod, toRod, fromRod);
}
int main() {
int numDisks;
// Input: Number of disks
printf("Enter the number of disks: ");
scanf("%d", &numDisks);
// Solve Tower of Hanoi
printf("The sequence of moves to solve Tower of Hanoi is:\n");
towerOfHanoi(numDisks, 'A', 'C', 'B');
return 0;
}
```

PROGRAM 5 REVERSE STRING POINTERS

```
#include <stdio.h>

#include <string.h>

void reverseString(char* str) {

char* start = str;

char* end = str + strlen(str) - 1;

char temp;

while (start < end) {

temp = *start;

*start = *end;

*end = temp;

start++;

end--;

}

}

int main() {

char str[100];

printf("Enter a string: ");

scanf("%s", str);

reverseString(str);

printf("Reversed string: %s\n", str);

return 0;

}
```

PROGRAM 6 ARRAY TO CALCULATE SUM AND AVGRAGE OF ELEMENTS

```
#include <stdio.h>

int main() {

int n, i;

float sum = 0, average;

int arr[100];

printf("Enter the number of elements: ");

scanf("%d", &n);

printf("Enter the elements: \n");

for (i = 0; i < n; i++) {

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

for (i = 0; i < n; i++) {

sum += arr[i];

}

average = sum / n;

printf("Sum of elements: %.2f\n", sum);

printf("Average of elements: %.2f\n", average);

return 0;

}
```

PROGRAM 7 LINEAR SEARCH BINARY SEARCH

```
#include <stdio.h>

void linearSearch(int arr[], int n, int target) {

int i;

for (i = 0; i < n; i++) {

if (arr[i] == target) {

printf("Element %d found at index %d\n", target, i);

return;

}

}

printf("Element %d not found\n", target);

}

void binarySearch(int arr[], int n, int target) {

int low = 0, high = n - 1, mid;

while (low <= high) {

mid = (low + high) / 2;

if (arr[mid] == target) {

printf("Element %d found at index %d\n", target, mid);

return;

} else if (arr[mid] < target)

low = mid + 1;

else

high = mid - 1;

}

printf("Element %d not found\n", target);

}
```

```
int main() {

int arr[100],n,choice,target,i,j,t;

clrscr();

printf("Enter the number of elements: ");

scanf("%d", &n);

printf("Enter the elements: \n");

for (i = 0; i < n; i++) {

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

}

printf("\n MENU \n");

printf("1. Linear Search\n");

printf("2. Binary Search\n");

scanf("%d", &choice);

printf("Enter the element to search for: ");

scanf("%d", &target);

switch (choice) {

case 1:

linearSearch(arr, n, target);

break;

case 2:

for (i = 0; i < n-1; i++) {

for (j = 0; j < n-i-1; j++) {

if (arr[j] > arr[j+1]) {

t = arr[j];

arr[j] = arr[j+1];

arr[j+1] = t;
}
}
}
binarySearch(arr, n, target);
break;
default:
printf("Invalid choice.\n");
}
getch();
return 0;
}
```

PROGRAM 8 QUICK SORT

```
#include <stdio.h>

#include <conio.h>

void swap(int *a, int *b) {

int temp;

temp = *a;

*a = *b;

*b = temp;

}

int partition(int arr[], int low, int high) {

int pivot = arr[high];

int j;

int i = low - 1;

for (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) {

int pi;

if (low < high) {

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++) {

printf("%d ", arr[i]);

}

printf("\n");

}

int main() {

int n,i;

int arr[100];

clrscr();

printf("Enter the number of elements: ");

scanf("%d", &n);

printf("Enter the elements: \n");

for (i = 0; i < n; i++) {

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

}

quickSort(arr, 0, n - 1);

printf("Sorted array: \n");

printArray(arr, n);

getch();

return 0;
```

```
}
```

PROGRAM 9 CIRCULAR QUEUE

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#define MAX 5
typedef struct {
int front, rear;
int arr[MAX];
}CircularQueue;
void initQueue(CircularQueue *q) {
q->front = -1;
q->rear = -1;
}
int isFull(CircularQueue *q) {
return ((q->rear + 1) % MAX == q->front);
}
int isEmpty(CircularQueue *q) {
return (q->front == -1);
}
void enqueue(CircularQueue *q, int value) {
if (isFull(q)) {
printf("Queue is full. Cannot enqueue %d\n", value);
} else {
if (q->front == -1) {
q->front = 0;
}
q->rear = (q->rear + 1) % MAX;
q->arr[q->rear] = value;
printf("%d enqueued to queue\n", value);
}
}
int dequeue(CircularQueue *q) {
if (isEmpty(q)) {
printf("Queue is empty. Cannot dequeue\n");
return -1;
} else {
int dequeuedValue = q->arr[q->front];
if (q->front == q->rear) {
q->front = q->rear = -1;
} else {
q->front = (q->front + 1) % MAX;
}
return dequeuedValue;
}
}
void display(CircularQueue *q) {
int i;
if (isEmpty(q)) {
printf("Queue is empty\n");
} else {
printf("Queue elements: ");
```

```
i = q->front;
while (i != q->rear) {
printf("%d ", q->arr[i]);
i = (i + 1) % MAX;
}
printf("%d\n", q->arr[q->rear]);
}
}
int main() {
CircularQueue q;
initQueue(&q);
enqueue(&q, 10);
enqueue(&q, 20);
enqueue(&q, 30);
enqueue(&q, 40);
enqueue(&q, 50);
enqueue(&q, 60);
display(&q);
printf("Dequeued: %d\n", dequeue(&q));
printf("Dequeued: %d\n", dequeue(&q));
enqueue(&q, 60);
enqueue(&q, 70);
display(&q);
getch();
return 0;
}
```