Assignment – I

1. Student Grade Management System

- Problem Statement: Create a program to manage student grades. Use:
    - A static variable to keep track of the total number of students processed.
    - A const global variable for the maximum number of grades.
    - A volatile variable to simulate an external grade update process.
    - Use if-else and switch to determine grades based on marks and a for loop to process multiple students.
- Key Concepts Covered: Storage classes (static, volatile), Type qualifiers (const), Decision-making (if-else, switch), Looping (for).

```
#include <stdio.h>
static int p=0;
const int g=4;
volatile int eg=0;

char grade(int m)
{
   if (m>=85)
      return 'A';
   else if(m>=65)
      return 'B';
   else if(m>=45)
      return 'C';
   else
      return 'F';
```

```
}

void main()
{
   int s,m,i;
   printf("Enter the number of students:\n");
   scanf("%d",&s);
   for(i=1;i<=s;i++)
   {
      eg=1;
      printf("Enter the marks of student %d: ",i);
      scanf("%d",&m);
      if(m<0 || m>100)
      {
         printf("Invalid marks\n");
         continue;
      }
      switch(grade(m))
      {
         case 'A':printf("A:Excellent\n");
                break;
         case 'B':printf("B:Good\n");
                break;
         case 'C':printf("C:Pass\n");
                break;
         case 'F':printf("F:Fail\n");
                break;
```

```
          default:printf("Not a valid grade\n");
        }
        p++;
      }
      printf("Total students processed: %d\n",p);
    }
```

O/P:

```
Enter the number of students:
5
Enter the marks of student 1: 58
C:Pass
Enter the marks of student 2: 90
A:Excellent
Enter the marks of student 3: 36
F:Fail
Enter the marks of student 4: 70
B:Good
Enter the marks of student 5: 102
Invalid marks
Total students processed: 4
```

2. Prime Number Finder

- Problem Statement: Write a program to find all prime numbers between 1 and a given number N. Use:
    - A const variable for the upper limit N.
    - A static variable to count the total number of prime numbers found.

    - Nested for loops for the prime-checking logic.

- Key Concepts Covered: Type qualifiers (const), Storage classes (static), Looping (for).

```
#include <stdio.h>
void main()
{
   static int c=0;
   int n,i;
   printf("Enter the upper limit N:\n");
   scanf("%d",&n);
   const int n1=n;
   printf("Prime numbers between 1 and %d are:\n",n1);
   for(n=1;n<=n1;n++)
   {
      for(i=2;i<n;i++)
      {
         if(n%i==0)
            break;
      }
      if(n==i)
      {
         printf("%d ",n);
         c++;
      }
   }
   printf("\nTotal number of prime numbers present: %d\n",c);
}
```

O/P:

```
Enter the upper limit N:
50
Prime numbers between 1 and 50 are:
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47
Total number of prime numbers present: 15
```

3. Dynamic Menu-Driven Calculator

- Problem Statement: Create a menu-driven calculator with options for addition, subtraction, multiplication, and division. Use:
  - A static variable to track the total number of operations performed.
  - A const pointer to hold operation names.
  - A do-while loop for the menu and a switch case for operation selection.
- Key Concepts Covered: Storage classes (static), Type qualifiers (const), Decision-making (switch), Looping (do-while).

```
#include <stdio.h>
void main()
{
    static int t=0;
    const char *op[]={"Addition", "Subtraction", "Multiplication", "Division"};
   int o,n1,n2,n3;
   do
   {
      printf("/nEnter the numbers n1 and n2:\n");
      scanf("%d %d",&n1,&n2);
```

```
            printf("Menu-Driven Calculator\n1. %s\n2. %s\n3. %s\n4.
%s\n",op[0],op[1],op[2],op[3]);
        printf("Enter the option:\n");
        scanf("%d",&o);
        switch(o)
        {
           case 1:n3=n1+n2;
                 printf("Addition of two numbers=%d\n",n3);
                 t++;
                 break;
           case 2:n3=n1-n2;
                 printf("Subtraction of two numbers=%d\n",n3);
                 t++;
                 break;
           case 3:n3=n1*n2;
                 printf("Multiplication of two numbers=%d\n",n3);
                 t++;
                 break;
           case 4:if(n2==0)
                 {
                    printf("Division by zero not allowed\n");
                    break;
                 }
                 else
                 {
                    n3=n1/n2;
                    printf("Division of two numbers=%d\n",n3);
                    t++;
```

```
                        break;
                    }
              default:printf("Invalid option\n");
                      break;
            }
            printf("Total operations: %d\n",t);
        } while(o!=4);
    }
```

O/P:

```
    Enter the numbers n1 and n2:
    6 5
    Menu-Driven Calculator
    1. Addition
    2. Subtraction
    3. Multiplication
    4. Division
    Enter the option:
    3
    Multiplication of two numbers=30
    Total operations: 1

    Enter the numbers n1 and n2:
    8 4
    Menu-Driven Calculator
    1. Addition
    2. Subtraction
```

```
3. Multiplication
4. Division
Enter the option:
1
Addition of two numbers=12
Total operations: 2

Enter the numbers n1 and n2:
0 6
Menu-Driven Calculator
1. Addition
2. Subtraction
3. Multiplication
4. Division
Enter the option:
2
Subtraction of two numbers=-6
Total operations: 3

Enter the numbers n1 and n2:
4 3
Menu-Driven Calculator
1. Addition
2. Subtraction
3. Multiplication
4. Division
Enter the option:
```

```
4
Division of two numbers=1
Total operations: 4

Enter the numbers n1 and n2:
6 0
Menu-Driven Calculator
1. Addition
2. Subtraction
3. Multiplication
4. Division
Enter the option:
4
Division by zero not allowed
Total operations: 0
```

4. Configuration-Based Matrix Operations

- Problem Statement: Perform matrix addition and multiplication. Use:
  - A const global variable to define the maximum size of the matrix.
  - static variables to hold intermediate results.
  - if statements to check for matrix compatibility.
  - Nested for loops for matrix calculations.
- Key Concepts Covered: Type qualifiers (const), Storage classes (static), Decision-making (if), Looping (nested for).

```
#include <stdio.h>
const int size=6;
```

```
static int res[6][6];

void add(int r, int c, int m1[size][size], int m2[size][size])
{
    int i,j;
    printf("\nAddition of 2 matrices:\n");
    for(i=0;i<r;i++)
    {
        for(j=0;j<c;j++)
            res[i][j]=m1[i][j]+m2[i][j];
    }
    printf("Resultant Matrix:\n");
    for(i=0;i<r;i++)
    {
        for(j=0;j<c;j++)
            printf("%d ",res[i][j]);
        printf("\n");
    }
}

void mul(int r1, int c1, int r2, int c2, int m1[size][size], int m2[size][size])
{
    if(c1!=r2) {
        printf("Multiplication of 2 matrices is not possible\n");
        return;
    }
    int i,j,k;
    printf("\nMultiplication of 2 matrices:\n");
    for(i=0;i<r1;i++)
    {
        for(j=0;j<c2;j++)
        {
            res[i][j]=0;
            for(k=0;k<c1;k++)
                res[i][j]+=m1[i][k]*m2[k][j];
        }
    }
    printf("Resultant Matrix:\n");
    for(i=0;i<r1;i++)
    {
        for(j=0;j<c2;j++)
            printf("%d ",res[i][j]);
        printf("\n");
    }
}

void main()
{
    int r1,c1,r2,c2,m1[size][size],m2[size][size],i,j;
    printf("Enter the number of rows and columns of first matrix:\n");
    scanf("%d %d",&r1,&c1);
    printf("Enter the number of rows and columns of second matrix:\n");
    scanf("%d %d",&r2,&c2);
    if(r1!=r2 || c1!=c2)
```

```
        printf("Addition of 2 matrices is not possible\n");
    else
    {
        printf("Enter the elements of first matrix:\n");
        for(i=0;i<r1;i++)
        {
            for(j=0;j<c1;j++)
                scanf("%d",&m1[i][j]);
        }
        printf("Enter the elements of second matrix:\n");
        for(i=0;i<r2;i++)
        {
            for(j=0;j<c2;j++)
                scanf("%d",&m2[i][j]);
        }
        add(r1,c1,m1,m2);
    }
    mul(r1,c1,r2,c2,m1,m2);
}
```

O/P:

```
Enter the number of rows and columns of first matrix:
3 3
Enter the number of rows and columns of second matrix:
3 3
Enter the elements of first matrix:
1 3 2
```

```
2 1 3
3 2 1
Enter the elements of second matrix:
1 2 3
3 1 2
2 3 1

Addition of 2 matrices:
Resultant Matrix:
2 5 5
5 2 5
5 5 2

Multiplication of 2 matrices:
Resultant Matrix:
14 11 11
11 14 11
11 11 14
```

5. Temperature Monitoring System

- Problem Statement: Simulate a temperature monitoring system using:
    - A volatile variable to simulate temperature input.
    - A static variable to hold the maximum temperature recorded.
    - if-else statements to issue warnings when the temperature exceeds thresholds.
    - A while loop to continuously monitor and update the temperature.

- Key Concepts Covered: Storage classes (volatile, static), Decision-making (if-else), Looping (while).

```
#include <stdio.h>
volatile float ct=0.0;
static float mt=-100.0;

void main()
{
   printf("Starting the temperature monitoring system\n");
   while(1)
   {
      printf("Enter the temperature:\n");
      scanf("%f",&ct);
      if(ct==-1)
      {
         printf("Stoping the temperature monitoring system\n");
         break;
      }
      if(ct>mt)
         mt=ct;
      if(ct<0)
         printf("Low temperature\n");
      else if(ct>80)
         printf("Temperature is too high\n");
      else if(ct>50)
         printf("Temperature is a little high\n");
      else
```

```
            printf("Temperature is normal\n");
         printf("Maximum Temperature Recorded: %.2f°C\n\n",mt);
      }
   }
```

O/P:

```
Starting the temperature monitoring system
Enter the temperature:
-6
Low temperature
Maximum Temperature Recorded: -6.00°C

Enter the temperature:
92
Temperature is too high
Maximum Temperature Recorded: 92.00°C

Enter the temperature:
64
Temperature is a little high
Maximum Temperature Recorded: 92.00°C

Enter the temperature:
33
Temperature is normal
Maximum Temperature Recorded: 92.00°C
```

```
Enter the temperature:
-1
Stoping the temperature monitoring system
```

6. Password Validator

- Problem Statement: Implement a password validation program. Use:
  - A static variable to count the number of failed attempts.
  - A const variable for the maximum allowed attempts.
  - if-else and switch statements to handle validation rules.
  - A do-while loop to retry password entry.
- Key Concepts Covered: Storage classes (static), Type qualifiers (const), Decision-making (if-else, switch), Looping (do-while).

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

const int a=3;
static int f=0;

void main()
{
   char p[50];
   int u=0,l=0,d=0,s=0,len=0,i;
   printf("Requirements for password:\n");
   printf("At least 8 characters(uppercase,lowercase,digit,special character at least one respectively)\n");
   do
```

```
{
    printf("Enter the password:\n");
    scanf("%s",p);
    for(i=0;p[i]!='\0';i++)
    {
        len++;
        switch (1)
        {
            case 1:if(isupper(p[i]))
                        u=1;
                  if(islower(p[i]))
                        l=1;
                  if(isdigit(p[i]))
                        d=1;
                  if(ispunct(p[i]))
                        s=1;
                  break;
            default:printf("Invalid characters\n");
                   break;
        }
    }
    if(len>=8 && u && l && d && s)
    {
        printf("Valid password\n");
        break;
    }
    else
```

```
        {
           printf("Invalid password\n");
           f++;
        }
        if(f>=a)
        {
           printf("Access denied\n");
           break;
        }
        printf("Attempts left: %d\n",a-f);
     } while(f<a);
  }
```

O/P:

```
Requirements for password:
At least 8 characters(uppercase,lowercase,digit,special character at least one respectively)
Enter the password:
harry78!
Invalid password
Attempts left: 2
Enter the password:
Potter@#7
Valid password
```

7. Bank Transaction Simulator

- Problem Statement: Simulate bank transactions. Use:

    - A static variable to maintain the account balance.
    - A const variable for the maximum withdrawal limit.
    - if-else statements to check transaction validity.
    - A do-while loop for performing multiple transactions.
- Key Concepts Covered: Storage classes (static), Type qualifiers (const), Decision-making (if-else), Looping (do-while).

```
#include <stdio.h>
static float bal=2000.0;
const float wl=1000.0;

void main()
{
   int op;
   float amt;
   printf("Initial Account Balance: %.2f",bal);
   do
   {
        printf("\nChoose the options:\n1. Deposit\n2. Withdraw\n3. Check Balance\n");
      printf("Enter the option:\n");
      scanf("%d",&op);
      switch(op)
      {
         case 1:printf("Enter amount to deposit: ");
               scanf("%f",&amt);
               if(amt>0)
               {
                    bal+=amt;
                    printf("Deposit successful, current Balance: %.2f\n",bal);
                }
                else
                    printf("Invalid deposit amount\n");
                break;
            case 2:printf("Enter amount to withdraw: ");
                scanf("%f",&amt);
                if(amt>bal)
                    printf("Failed, insufficient balance\n");
                else if(amt>wl)
                    printf("Amount exceeds the maximum withdrawal limit of
%.2f\n",wl);
                else if(amt<=0)
                    printf("Invalid withdrawal amount\n");
                else
                {
                    bal-=amt;
                   printf("Withdrawal successful, current balance: %.2f\n",bal);
                }
                break;
            case 3:printf("Current Account Balance: %.2f\n",bal);
                break;
            default:printf("Invalid option\n");
                 break;
        }
    } while(op!=3);
}
```

O/P:

```
Initial Account Balance: 2000.00
Choose the options:
1. Deposit
2. Withdraw
3. Check Balance
Enter the option:
1
Enter amount to deposit: 200
Deposit successful, current Balance: 2200.00

Choose the options:
1. Deposit
2. Withdraw
3. Check Balance
Enter the option:
2
Enter amount to withdraw: 1000
Withdrawal successful, current balance: 1200.00

Choose the options:
1. Deposit
2. Withdraw
3. Check Balance
Enter the option:
3
```

```
Current Account Balance: 1200.00
```

8. Digital Clock Simulation

- Problem Statement: Simulate a digital clock. Use:
  - volatile variables to simulate clock ticks.
  - A static variable to count the total number of ticks.
  - Nested for loops for hours, minutes, and seconds.
  - if statements to reset counters at appropriate limits.
- Key Concepts Covered: Storage classes (volatile, static), Decision-making (if), Looping (nested for).

```
#include <stdio.h>
#include <unistd.h>
volatile int t=0;
static int total_t=0;

void main()
{
  int h=0,m=0,s=0;
  printf("HH:MM:SS\n");
  for(h=0;h<24;h++)  //for(h=0;h<1;h++)
  {
    for(m=0;m<60;m++)  //for(m=0;m<1;m++)
    {
      for(s=0;s<60;s++)
      {
        t=1;
```

```
                total_t++;
                printf("%02d:%02d:%02d\n",h,m,s);
                sleep(1);
                t=0;
            }
        }
    }
    printf("Total ticks: %d\n",total_t);
}
```

O/P:

```
HH:MM:SS
00:00:00
00:00:01
00:00:02
00:00:03
00:00:04
00:00:05
00:00:06
00:00:07
00:00:08
00:00:09
00:00:10
00:00:11
00:00:12
00:00:13
00:00:14
```

00:00:15

00:00:16

00:00:17

00:00:18

00:00:19

00:00:20

00:00:21

00:00:22

00:00:23

00:00:24

00:00:25

00:00:26

00:00:27

00:00:28

00:00:29

00:00:30

00:00:31

00:00:32

00:00:33

00:00:34

00:00:35

00:00:36

00:00:37

00:00:38

00:00:39

00:00:40

00:00:41

```
00:00:42
00:00:43
00:00:44
00:00:45
00:00:46
00:00:47
00:00:48
00:00:49
00:00:50
00:00:51
00:00:52
00:00:53
00:00:54
00:00:55
00:00:56
00:00:57
00:00:58
00:00:59
Total ticks: 60
```

9. Game Score Tracker

- Problem Statement: Track scores in a simple game. Use:
    - A static variable to maintain the current score.
    - A const variable for the winning score.
    - if-else statements to decide if the player has won or lost.
    - A while loop to play rounds of the game.

- Key Concepts Covered: Storage classes (static), Type qualifiers (const), Decision-making (if-else), Looping (while).

```
#include <stdio.h>
static int s=0;
const int ws=8;

void main()
{
   int p,r=1;
   printf("%d points to win the game\n",ws);
   while(s<ws)
   {
      printf("Round %d: Enter the points: ",r);
      scanf("%d",&p);
      if(p<0)
      {
         printf("Negative points not allowed\n");
         continue;
      }
      s+=p;
      printf("Current Score: %d\n",s);
      if(s>=ws)
      {
         printf("Game won with %d points\n",s);
         break;
      }
      r++;
```

```
    }
}
```

O/P:

```
8 points to win the game
Round 1: Enter the points: 5
Current Score: 5
Round 2: Enter the points: 2
Current Score: 7
Round 3: Enter the points: 3
Current Score: 10
Game won with 10 points
```