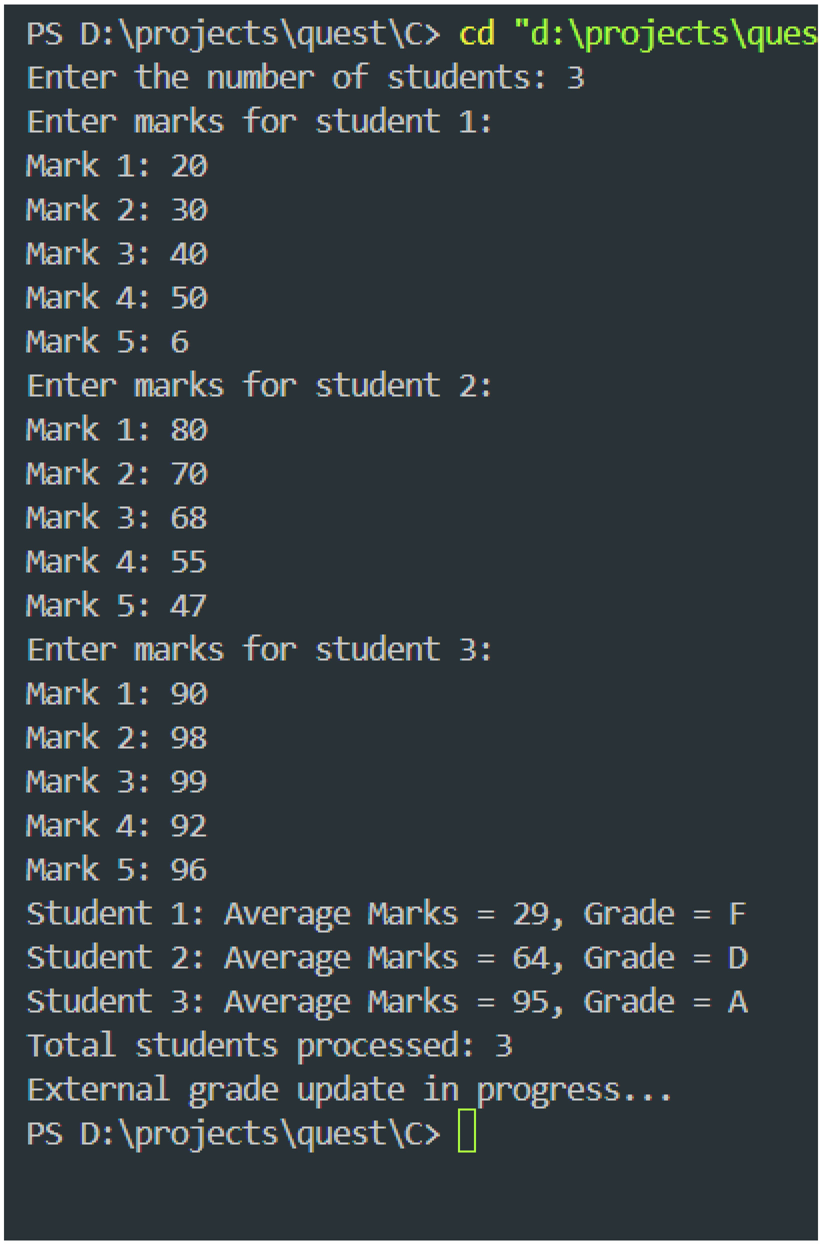
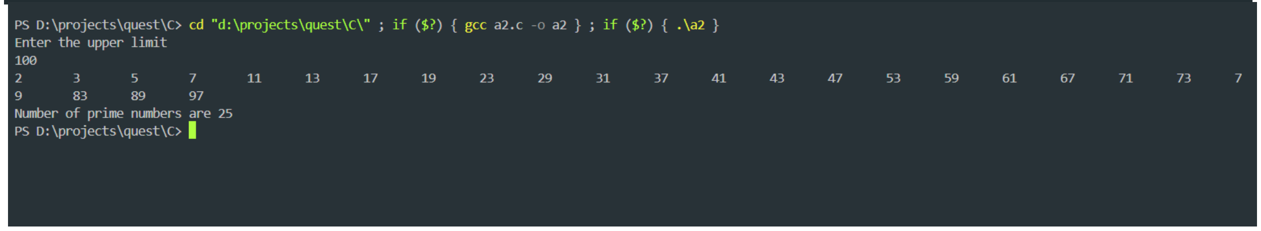
```c
/*
Student Grade Management System
Problem Statement: Create a program to manage student grades. Use:
A static variable to keep track of the total number of students processed.
A const global variable for the maximum number of grades.
A volatile variable to simulate an external grade update process.
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>
const int MAX_GRADES = 5;
static int total_students = 0;
volatile int external_grade_update = 0;
char determine_grade(int marks) {
    char grade;

    if (marks >= 90) {
        grade = 'A';
    } else if (marks >= 80) {
        grade = 'B';
    } else if (marks >= 70) {
        grade = 'C';
    } else if (marks >= 60) {
        grade = 'D';
    } else {
        grade = 'F';
    }

    return grade;
}


int main() {
    int num_students, i;
    printf("Enter the number of students: ");
    scanf("%d", &num_students);
    int student_marks[num_students][MAX_GRADES];
    for (i = 0; i < num_students; i++) {
        printf("Enter marks for student %d:\n", i + 1);
        for (int j = 0; j < MAX_GRADES; j++) {
            printf("Mark %d: ", j + 1);
            scanf("%d", &student_marks[i][j]);
        }

        total_students++;
    }
    for (i = 0; i < num_students; i++) {
        int total_marks = 0;
        for (int j = 0; j < MAX_GRADES; j++) {
            total_marks += student_marks[i][j];
        }

        int average_marks = total_marks / MAX_GRADES;
        char grade = determine_grade(average_marks);

        printf("Student %d: Average Marks = %d, Grade = %c\n", i + 1,
average_marks, grade);
    }

    printf("Total students processed: %d\n", total_students);

    external_grade_update = 1;
    if (external_grade_update) {
        printf("External grade update in progress...\n");
    }

    return 0;
}
```



```
PS D:\projects\quest\C> cd "d:\projects\ques
Enter the number of students: 3
Enter marks for student 1:
Mark 1: 20
Mark 2: 30
Mark 3: 40
Mark 4: 50
Mark 5: 6
Enter marks for student 2:
Mark 1: 80
Mark 2: 70
Mark 3: 68
Mark 4: 55
Mark 5: 47
Enter marks for student 3:
Mark 1: 90
Mark 2: 98
Mark 3: 99
Mark 4: 92
Mark 5: 96
Student 1: Average Marks = 29, Grade = F
Student 2: Average Marks = 64, Grade = D
Student 3: Average Marks = 95, Grade = A
Total students processed: 3
External grade update in progress...
PS D:\projects\quest\C>
```

```
/*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>
const int N;
```

```c
static int count=0;
void main(){
    printf("Enter the upper limit\n");
    scanf("%d",&N);
    for (int i = 2; i <= N; i++)
    {
        int is_prime = 1;
        for (int j = 2; j <= i/2; j++)
        {
            if(i%j == 0)
            {
                is_prime = 0;
                break;

            }
        }
        if(is_prime)
        {
            printf("%d \t",i);
            count++;
        }


    }
    printf("\nNumber of prime numbers are %d \n",count);

}
```

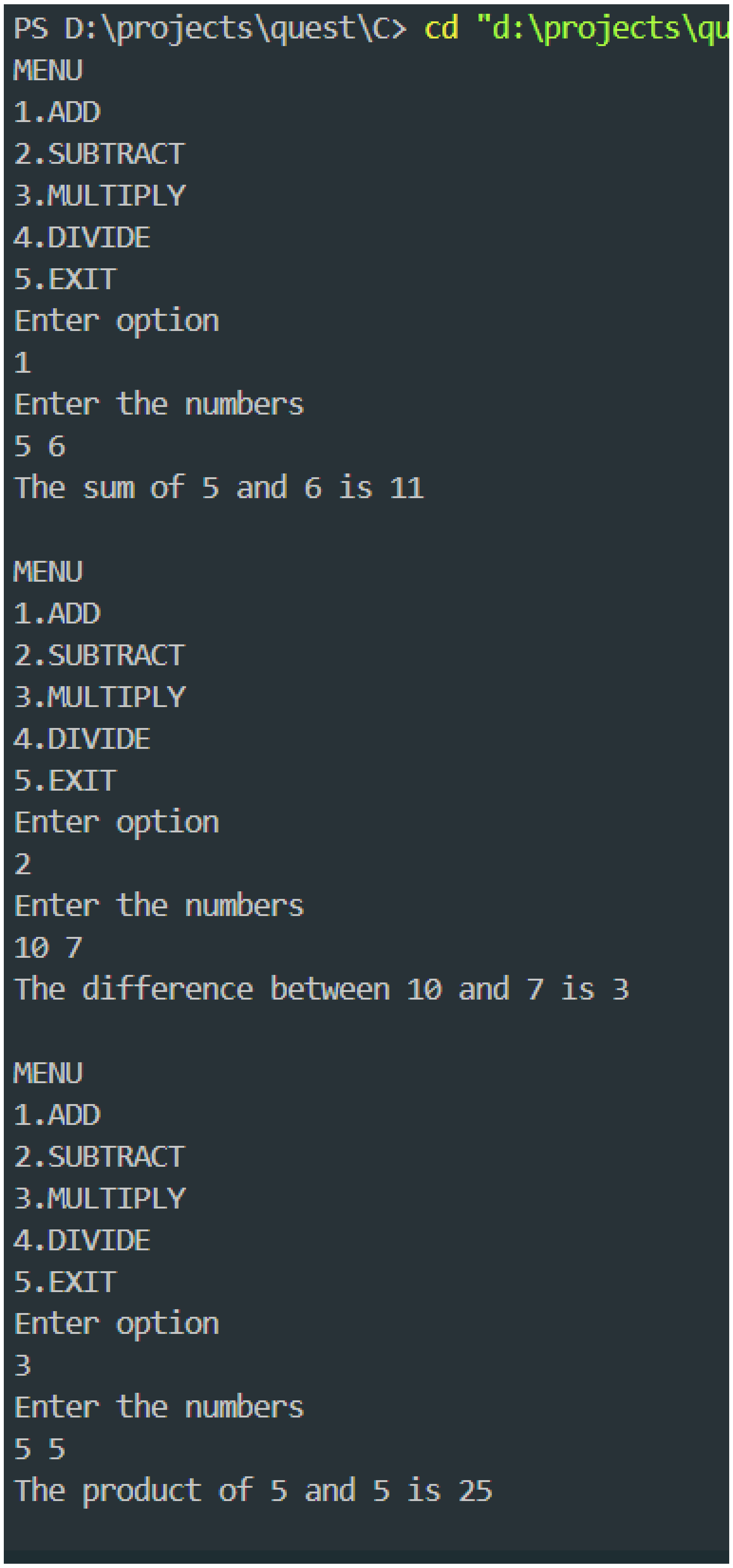


```
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ; if ($?) { gcc a2.c -o a2 } ; if ($?) { .\a2 }
Enter the upper limit
100
2       3       5       7       11      13      17      19      23      29      31      37      41      43      47      53      59      61      67      71      73      7
9       83      89      97
Number of prime numbers are 25
PS D:\projects\quest\C>
```

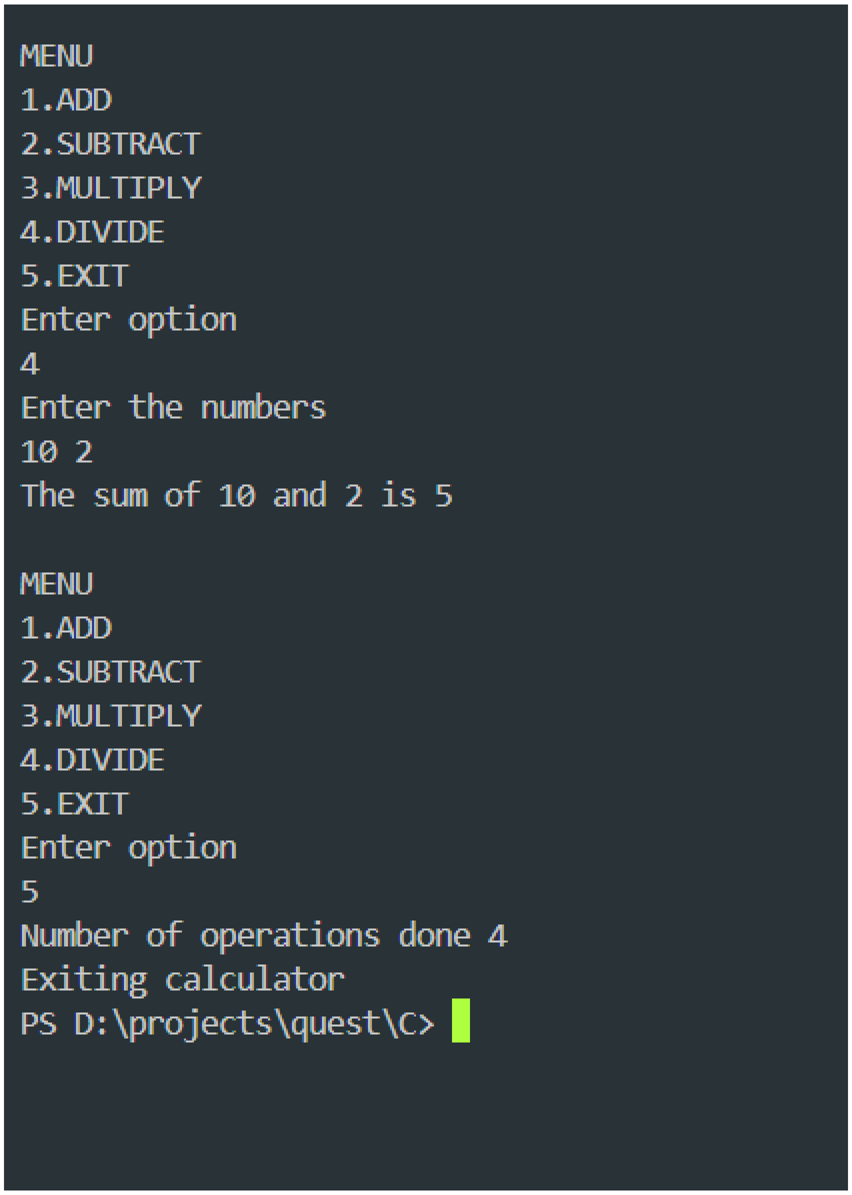
```c
/*Dynamic Menu-Driven Calculator
Problem Statement: Create a menu-driven calculator with options for
addition, subtraction, multiplication, and division. Use:
A static variable to track the total number of operations performed.
A const pointer to hold operation names.
A do-while loop for the menu 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>
#include<stdlib.h>
void add()
{
    int num1,num2;
    printf("Enter the numbers\n");
    scanf("%d %d",&num1,&num2);
    printf("The sum of %d and %d is %d\n",num1,num2,num1+num2);
    printf("\n");
}
void subtract()
{
    int num1,num2;
    printf("Enter the numbers\n");
    scanf("%d %d",&num1,&num2);
    printf("The difference between %d and %d is
%d\n",num1,num2,num1-num2);
    printf("\n");
}
void multiply()
{
    int num1,num2;
    printf("Enter the numbers\n");
    scanf("%d %d",&num1,&num2);
    printf("The product of %d and %d is %d\n",num1,num2,num1*num2);
    printf("\n");
}
void divide()
{
    int num1,num2;
    printf("Enter the numbers\n");
    scanf("%d %d",&num1,&num2);
    if(num2!=0)
    printf("The sum of %d and %d is %d\n",num1,num2,num1/num2);
    else
    printf("Division by zero is not possible\n");
    printf("\n");
}
void main(){
```

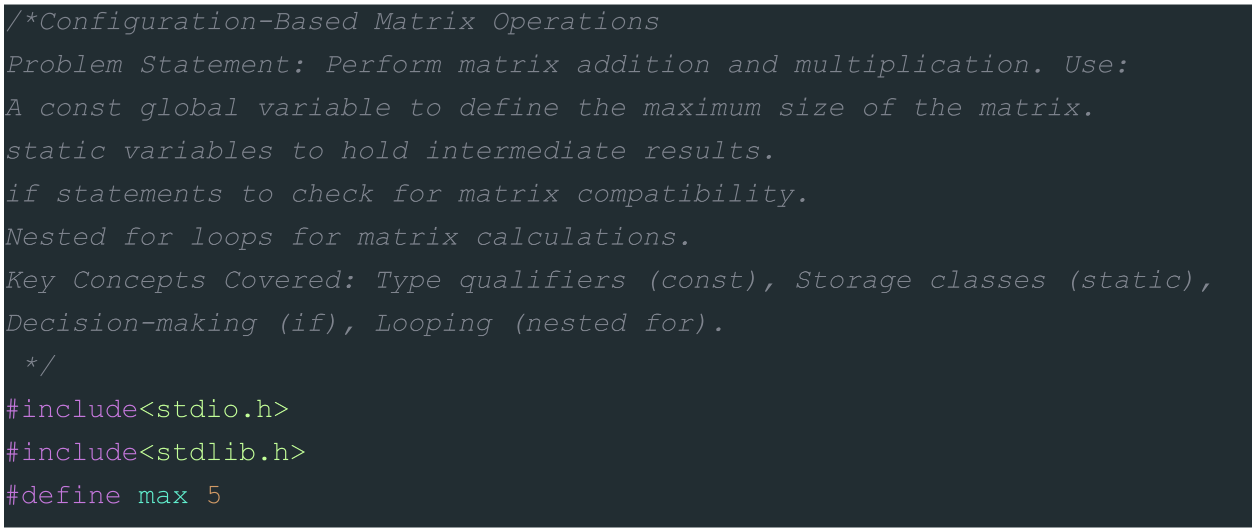
```c
    static int ops=0;
    const char *opertion[]={"ADD","SUBTRACT","MULTIPLY","DIVIDE"};
    int choice,num1,num2;
do
{
    printf("MENU\n");
    printf("1.%s\n",opertion[0]);
    printf("2.%s\n",opertion[1]);
    printf("3.%s\n",opertion[2]);
    printf("4.%s\n",opertion[3]);
    printf("5.EXIT\n");
    printf("Enter option\n");
    scanf("%d",&choice);

    switch(choice) {
    case 1 :add();
        ops++;
        break;
    case 2: subtract();
        ops++;
        break;
    case 3: multiply();
        ops++;
        break;
    case 4: divide();
        ops++;
        break;
    case 5:
        printf("Number of operations done %d\n",ops);
        printf("Exiting calculator\n");
        break;

    default: printf("Enter a valid option\n");
        break;
    }

} while (choice != 5);
}
```



```
PS D:\projects\quest\C> cd "d:\projects\qu
MENU
1.ADD
2.SUBTRACT
3.MULTIPLY
4.DIVIDE
5.EXIT
Enter option
1
Enter the numbers
5 6
The sum of 5 and 6 is 11

MENU
1.ADD
2.SUBTRACT
3.MULTIPLY
4.DIVIDE
5.EXIT
Enter option
2
Enter the numbers
10 7
The difference between 10 and 7 is 3

MENU
1.ADD
2.SUBTRACT
3.MULTIPLY
4.DIVIDE
5.EXIT
Enter option
3
Enter the numbers
5 5
The product of 5 and 5 is 25
```

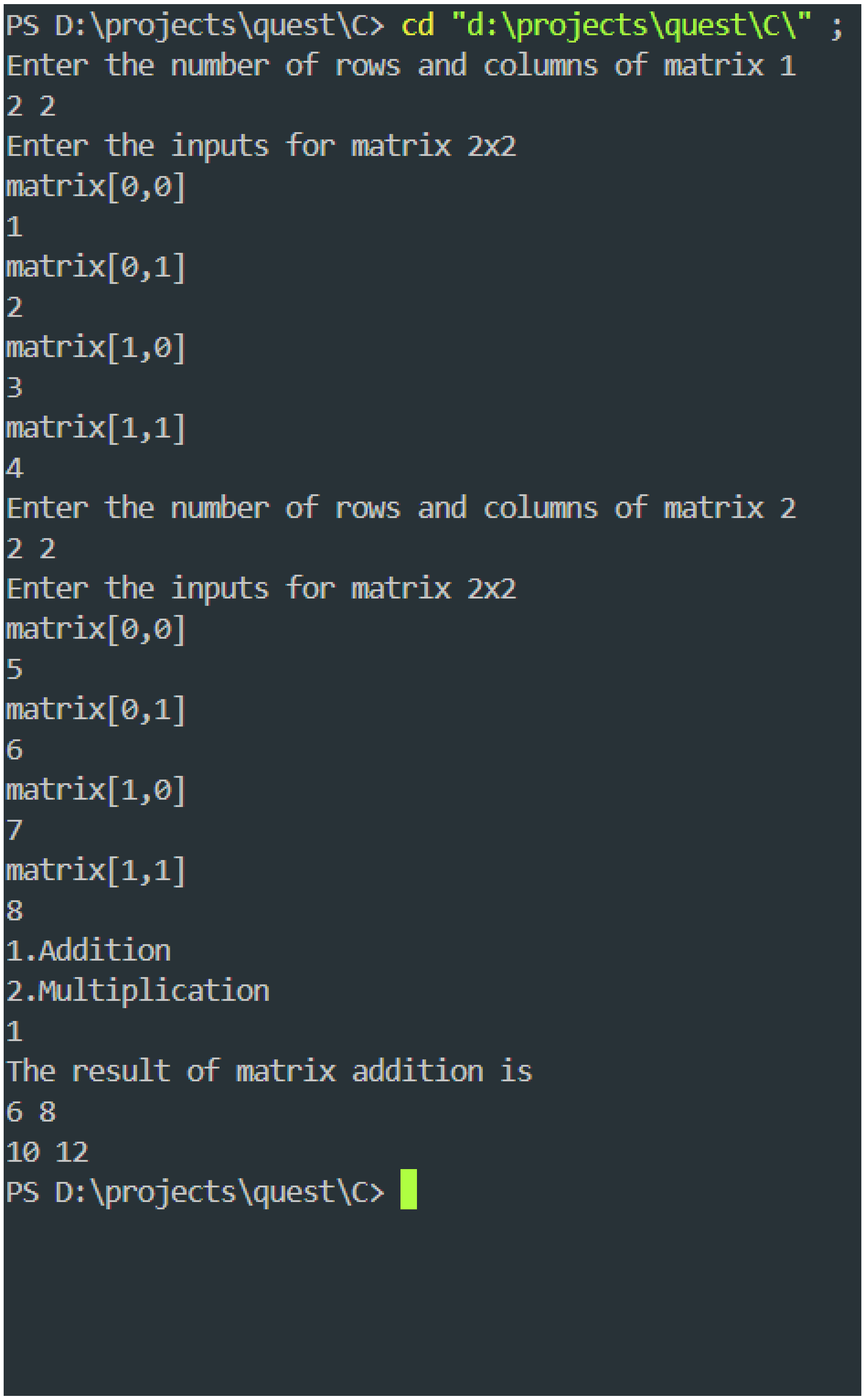


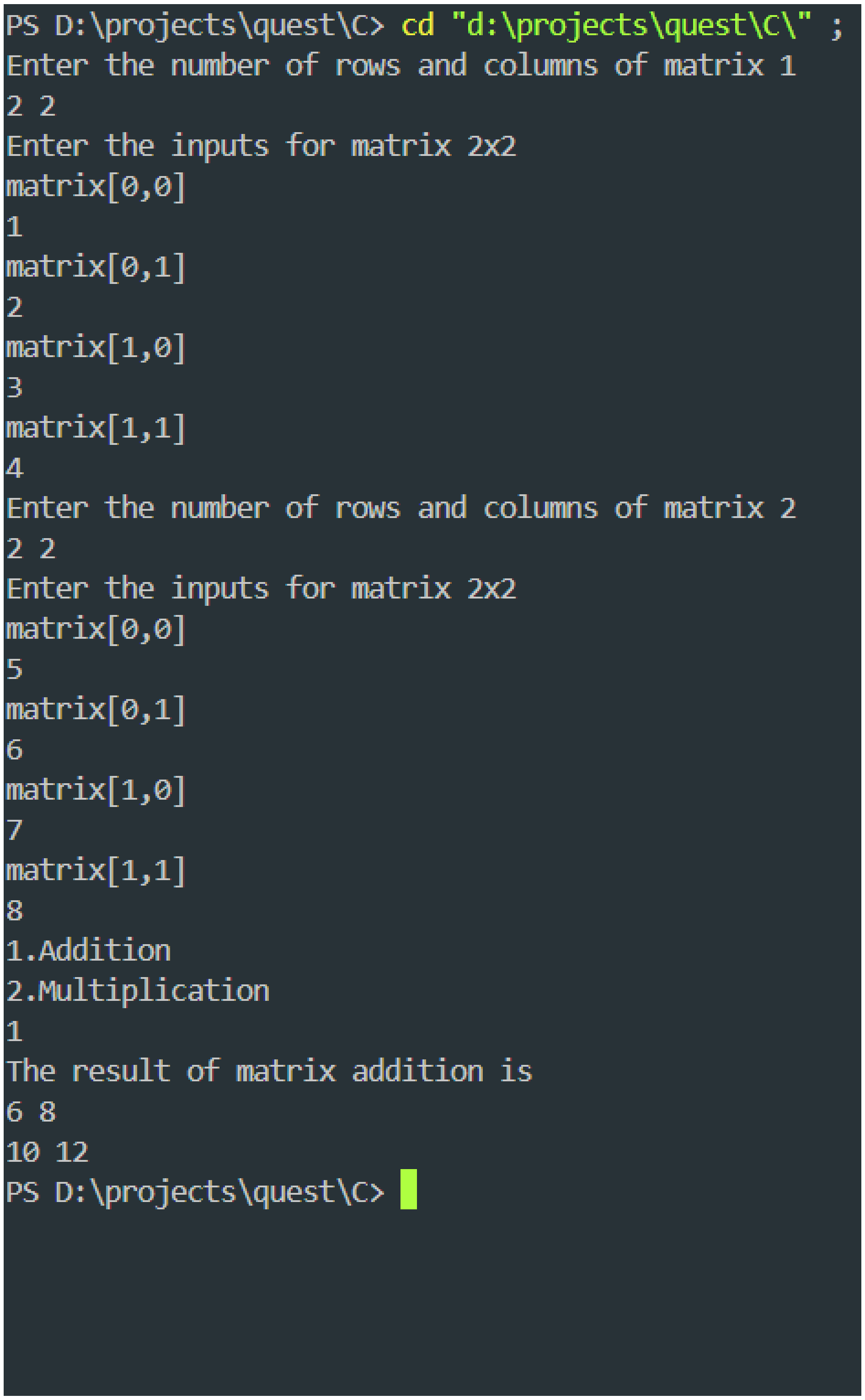
```
MENU
1.ADD
2.SUBTRACT
3.MULTIPLY
4.DIVIDE
5.EXIT
Enter option
4
Enter the numbers
10 2
The sum of 10 and 2 is 5

MENU
1.ADD
2.SUBTRACT
3.MULTIPLY
4.DIVIDE
5.EXIT
Enter option
5
Number of operations done 4
Exiting calculator
PS D:\projects\quest\C>
```



```c
/*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>
#include<stdlib.h>
#define max 5
```

```c
const int max_size=max;
void input(int m[max][max],int r,int c)
{
    printf("Enter the inputs for matrix %dx%d\n",r,c);
    for(int i=0;i<r;i++)
    {
        for(int j=0;j<c;j++)
        {
            printf("matrix[%d,%d]\n",i,j);
            scanf("%d",&m[i][j]);
        }
    }
}
void display(int m[max][max],int r,int c)
{
    for(int i=0;i<r;i++)
    {
        for(int j=0;j<c;j++)
        {
            printf("%d ",m[i][j]);
        }
        printf("\n");
    }
}
void add(int m1[max][max],int m2[max][max],int result[max][max],int
row,int col)
{
    for (int i = 0; i < row; i++) {
        for (int j = 0; j < col; j++) {
            result[i][j] = m1[i][j] + m2[i][j];
        }
    }
}
void multiply(int m1[max][max],int m2[max][max],int result[max][max],int
row1,int col1,int row2,int col2) {
    for (int i = 0; i < row1; i++) {
        for (int j = 0; j < col2; j++) {
            result[i][j] = 0;
        }
    }
    for (int i = 0; i < row1; i++) {
        for (int j = 0; j < col2; j++) {
            for (int k = 0; k < col1; k++) {
                result[i][j] += m1[i][k] * m2[k][j];
            }
        }
    }
}
void main()
{
    static int m1[max][max],m2[max][max],result[max][max];
    int row1,col1,row2,col2,choice;
    printf("Enter the number of rows and columns of matrix 1\n");
    scanf("%d %d",&row1,&col1);
    if(row1>max_size || col1>max_size)
    {   printf("Dimension greater than max size\n");
        exit(0);
    }
    else
    {
        input(m1,row1,col1);
    }
    printf("Enter the number of rows and columns of matrix 2\n");
    scanf("%d %d",&row2,&col2);
    if(row2>max_size || col2>max_size)
    {   printf("Dimension greater than max size\n");
        exit(0);
    }
     else
    {
        input(m2,row2,col2);
    }
    printf("1.Addition\n");
    printf("2.Multiplication\n");
    scanf("%d",&choice);
    if(choice==1)
    {
    if(row1==row2 &&col1 == col2)
        {
            add(m1,m2,result,row1,col1);
            printf("The result of matrix addition is \n");
            display(result,row1,col1);
        }
     else
            printf("Incompatible dimension\n");
    }
    else if(choice ==2)
    {
        if(col1 == row2)
        {
            multiply(m1,m2,result,row1,col1,row2,col2);
            printf("The result of matrix multiplication is \n");
            display(result,row1,col2);
        }
        else
            printf("Incompatible dimension\n");
    }
}
```

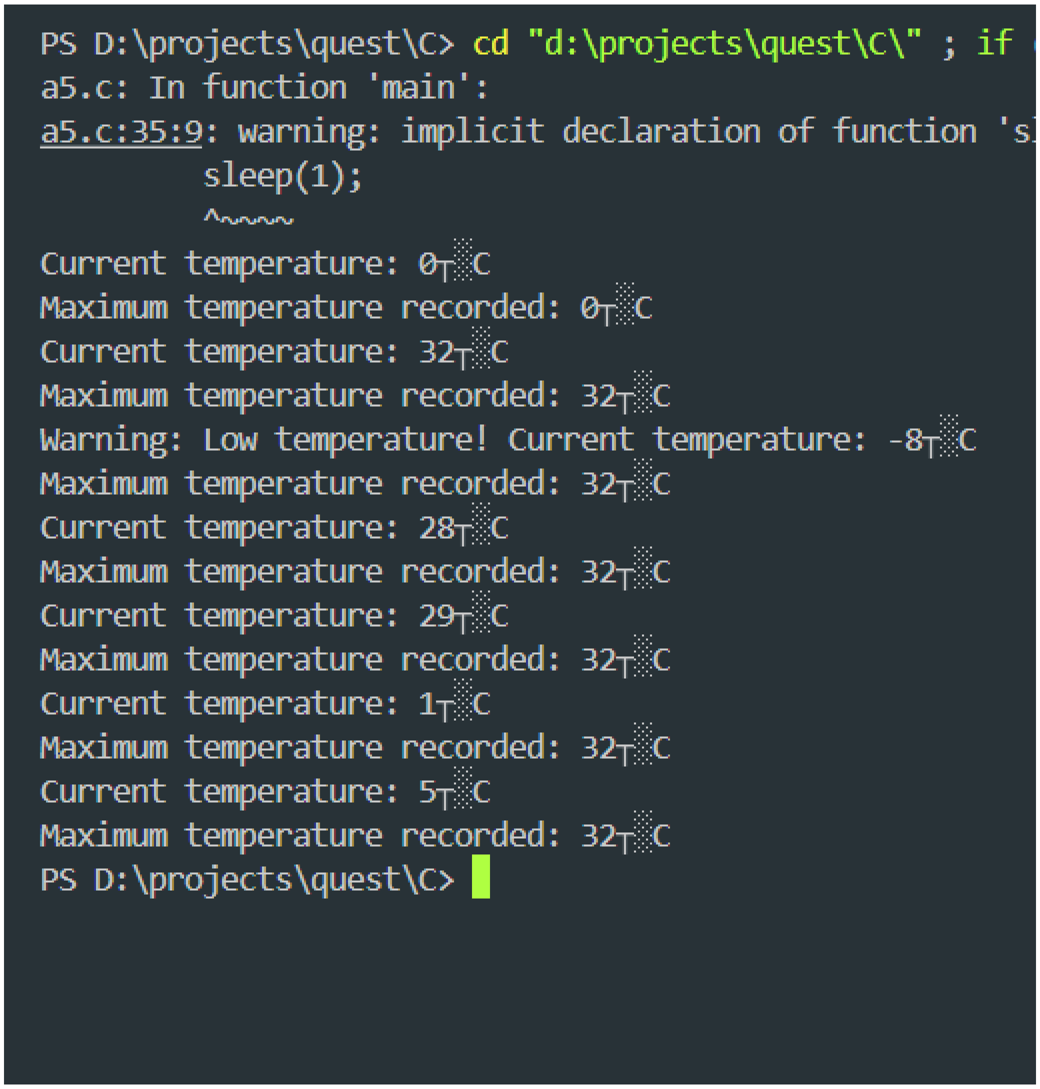

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ;
Enter the number of rows and columns of matrix 1
2 2
Enter the inputs for matrix 2x2
matrix[0,0]
1
matrix[0,1]
2
matrix[1,0]
3
matrix[1,1]
4
Enter the number of rows and columns of matrix 2
2 2
Enter the inputs for matrix 2x2
matrix[0,0]
5
matrix[0,1]
6
matrix[1,0]
7
matrix[1,1]
8
1.Addition
2.Multiplication
1
The result of matrix addition is
6 8
10 12
PS D:\projects\quest\C>
```



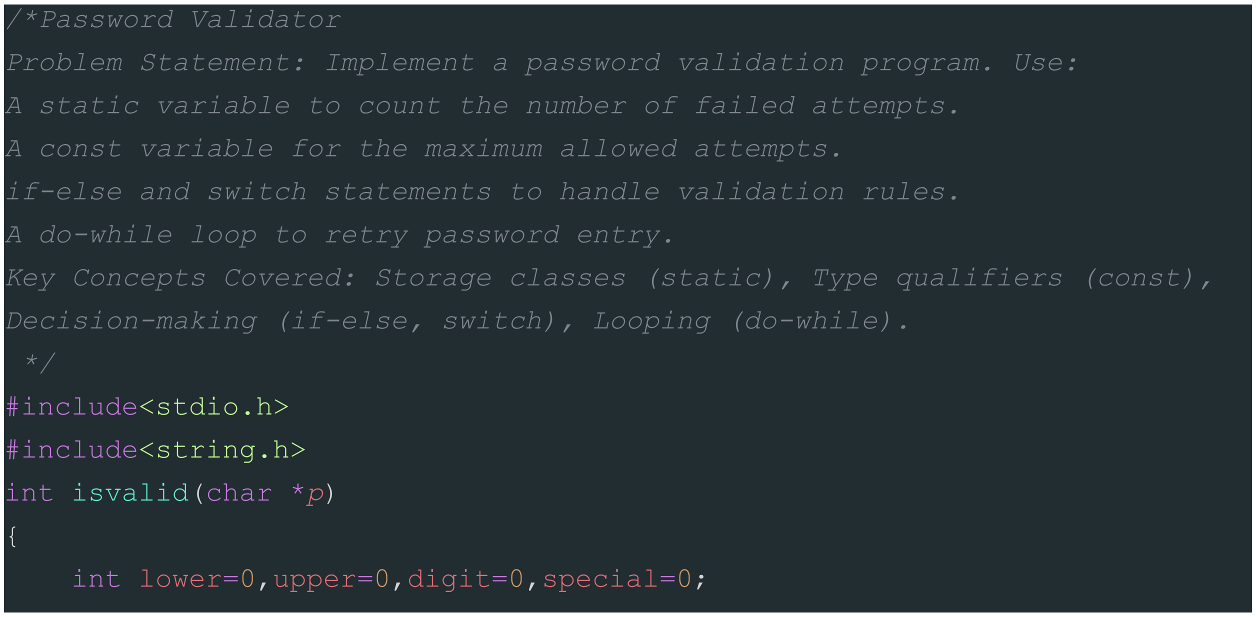
```
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ;
Enter the number of rows and columns of matrix 1
2 2
Enter the inputs for matrix 2x2
matrix[0,0]
1
matrix[0,1]
2
matrix[1,0]
3
matrix[1,1]
4
Enter the number of rows and columns of matrix 2
2 2
Enter the inputs for matrix 2x2
matrix[0,0]
5
matrix[0,1]
6
matrix[1,0]
7
matrix[1,1]
8
1.Addition
2.Multiplication
1
The result of matrix addition is
6 8
10 12
PS D:\projects\quest\C> 
```

```
/*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.
```

```c
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>
#include <stdlib.h>
#include <time.h>
volatile int temperature = 0;
static int max_temperature = -273;
void updateTemperature() {
    temperature = (rand() % 71) - 20;
}

void main()
{
    srand(time(0));
    while (1) {
        updateTemperature();
        if (temperature > max_temperature) {
            max_temperature = temperature;
        }
        if (temperature > 40) {
            printf("Warning: High temperature! Current temperature:
%d°C\n", temperature);
        } else if (temperature < 0) {
            printf("Warning: Low temperature! Current temperature:
%d°C\n", temperature);
        } else {
            printf("Current temperature: %d°C\n", temperature);
        }

        printf("Maximum temperature recorded: %d°C\n", max_temperature);
        sleep(1);
    }
}
```



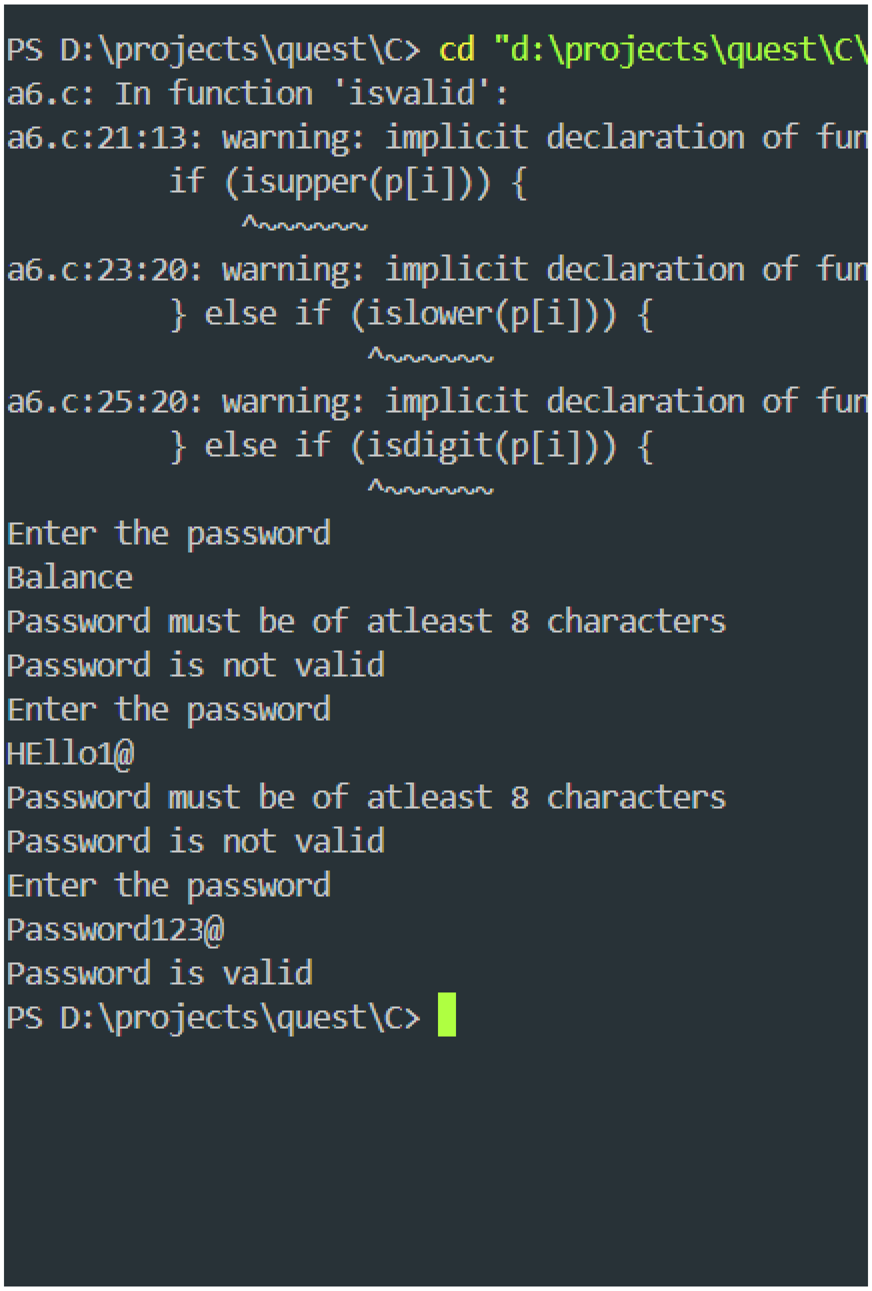
```
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ; if
a5.c: In function 'main':
a5.c:35:9: warning: implicit declaration of function 's
         sleep(1);
         ^~~~~
Current temperature: 0┬░C
Maximum temperature recorded: 0┬░C
Current temperature: 32┬░C
Maximum temperature recorded: 32┬░C
Warning: Low temperature! Current temperature: -8┬░C
Maximum temperature recorded: 32┬░C
Current temperature: 28┬░C
Maximum temperature recorded: 32┬░C
Current temperature: 29┬░C
Maximum temperature recorded: 32┬░C
Current temperature: 1┬░C
Maximum temperature recorded: 32┬░C
Current temperature: 5┬░C
Maximum temperature recorded: 32┬░C
PS D:\projects\quest\C>
```



```
/*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>
int isvalid(char *p)
{
    int lower=0,upper=0,digit=0,special=0;
```

```c
    int len =strlen(p);
    if(len<8)
    {
        printf("Password must be of atleast 8 characters\n");
        return 0;
    }
    for (int i = 0; i < len; i++) {
        if (isupper(p[i])) {
            upper = 1;
        } else if (islower(p[i])) {
            lower = 1;
        } else if (isdigit(p[i])) {
            digit = 1;
        } else {
            special = 1;
        }
    }
    if(!lower)
    {
        printf("Does not contain atlest one lowercase\n");
        return 0;
    }
    if(!upper)
    {
         printf("Does not contain atlest one uppercase\n");
        return 0;
    }
    if(!digit)
    {
        printf("Does not contain atlest one digit\n");
        return 0;
    }
    if(!special)
    {
        printf("Does not contain atlest one special character\n");
        return 0;
    }
    return 1;

}
```

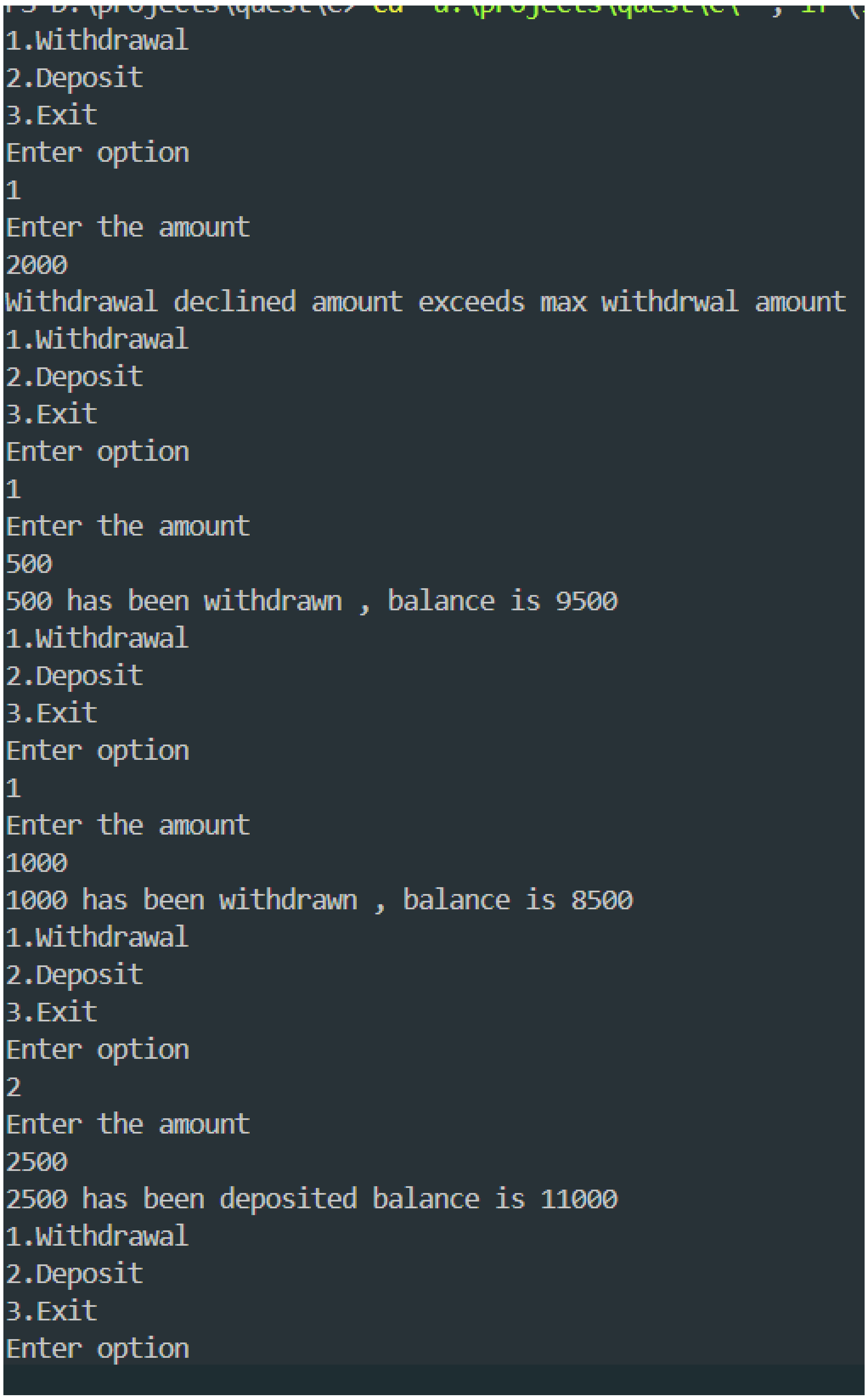
```c
void main()
{
    const int max=5;
    static int fail=0;
    char p[100];
    int valid=0;
   do
   {
    printf("Enter the password\n");
    scanf("%s",&p);
    valid=isvalid(p);
    if(valid)
    {
        printf("Password is valid\n");
        break;
    }
    else
    {
        printf("Password is not valid\n");
        fail++;
    }

   } while (fail<max);
   if(fail>=max)
   printf("Exceeded maximum number of attempts\n");

}
```



```
PS D:\projects\quest\C> cd "d:\projects\quest\C\
a6.c: In function 'isvalid':
a6.c:21:13: warning: implicit declaration of fun
         if (isupper(p[i])) {
             ^~~~~~~
a6.c:23:20: warning: implicit declaration of fun
         } else if (islower(p[i])) {
                    ^~~~~~~
a6.c:25:20: warning: implicit declaration of fun
         } else if (isdigit(p[i])) {
                    ^~~~~~~
Enter the password
Balance
Password must be of atleast 8 characters
Password is not valid
Enter the password
HEllo1@
Password must be of atleast 8 characters
Password is not valid
Enter the password
Password123@
Password is valid
PS D:\projects\quest\C>
```

```
/*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>
```

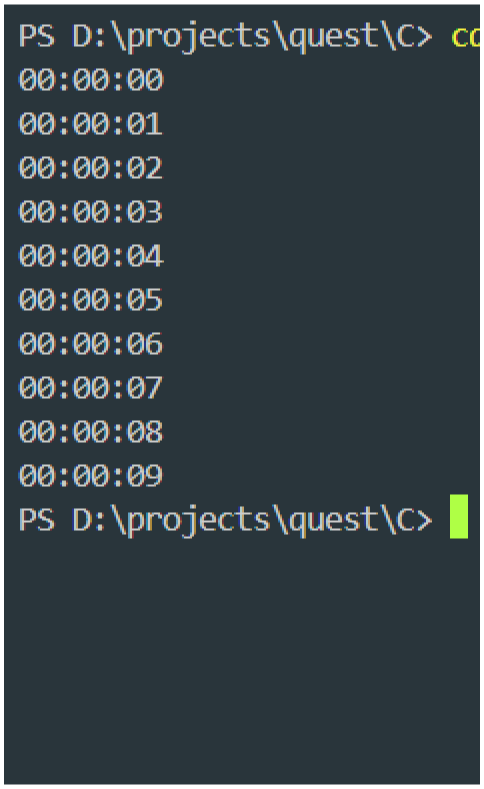
```
void main()
{
    static int balance = 10000;
    const int max = 1000;
    int deposit=0,withdrawl=0,choice;
    do
    {
        printf("1.Withdrawal\n2.Deposit\n3.Exit\n");
        printf("Enter option\n");
        scanf("%d",&choice);
        if(choice == 1)
        {
            printf("Enter the amount\n");
            scanf("%d",&withdrawl);
            if(withdrawl > max)
            {
                printf("Withdrawal declined amount exceeds max withdrwal
amount\n");
            }
            else
            {
                if(withdrawl>balance)
                printf("Withdrawal declined amount exceeds balance
amount\n");
                else
                {
                    balance = balance - withdrawl;
                    printf("%d has been withdrawn , balance is
%d\n",withdrawl,balance);
                }
            }
        }
        else if(choice ==2)
        {
            printf("Enter the amount\n");
            scanf("%d",&deposit);
            if(deposit > 0)
            {
                balance+=deposit;
                printf("%d has been deposited balance is 
%d\n",deposit,balance);
            }
            else
            {
                printf("Only positive amounts can be deposited");
            }
        }
        else
        break;
    } while (1);

}
```



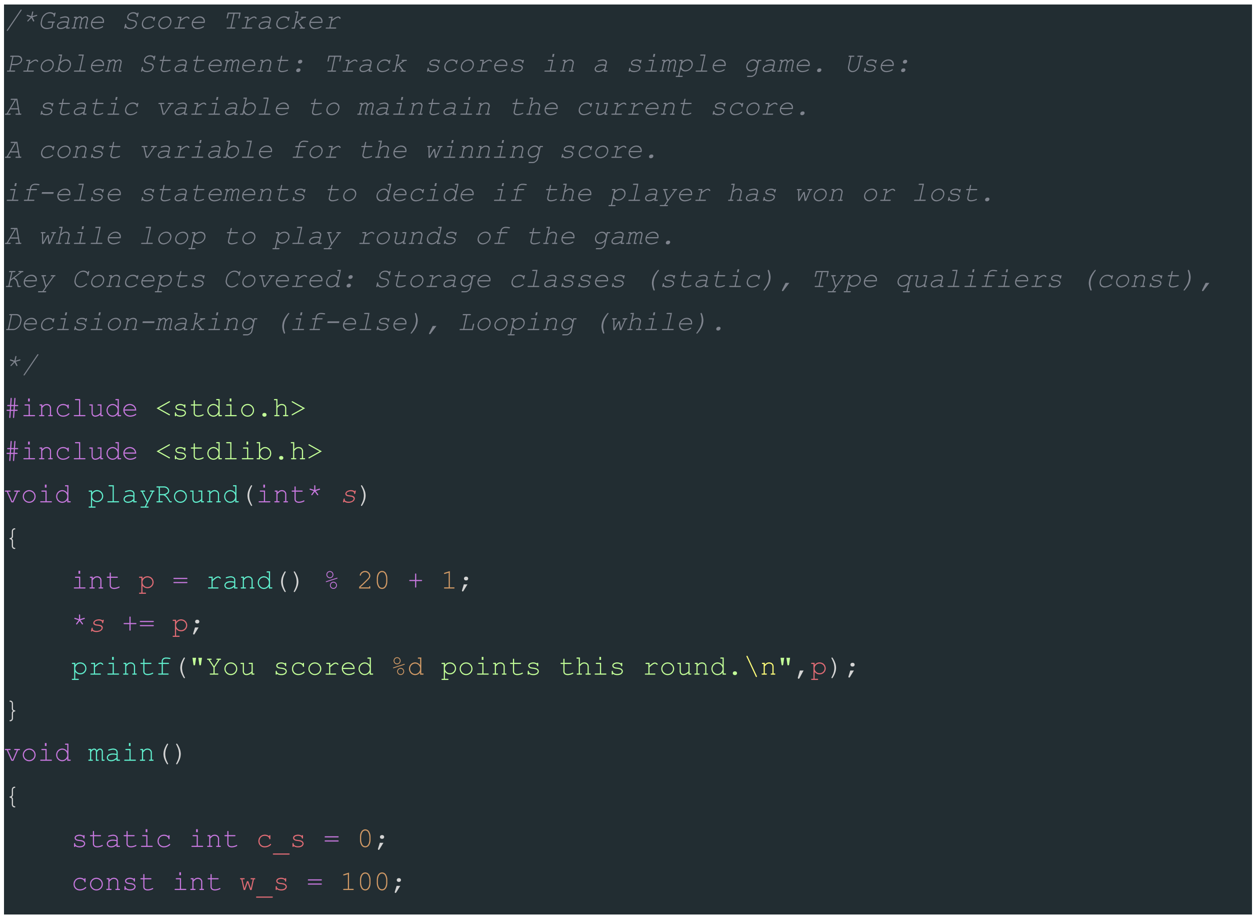
```
PS D:\projects\quest\c> cd "d:\projects\quest\c\" ; if (
1.Withdrawal
2.Deposit
3.Exit
Enter option
1
Enter the amount
2000
Withdrawal declined amount exceeds max withdrwal amount
1.Withdrawal
2.Deposit
3.Exit
Enter option
1
Enter the amount
500
500 has been withdrawn , balance is 9500
1.Withdrawal
2.Deposit
3.Exit
Enter option
1
Enter the amount
1000
1000 has been withdrawn , balance is 8500
1.Withdrawal
2.Deposit
3.Exit
Enter option
2
Enter the amount
2500
2500 has been deposited balance is 11000
1.Withdrawal
2.Deposit
3.Exit
Enter option
```

```
/*Digital Clock Simulation
Problem Statement: Simulate a digital clock. Use:
```

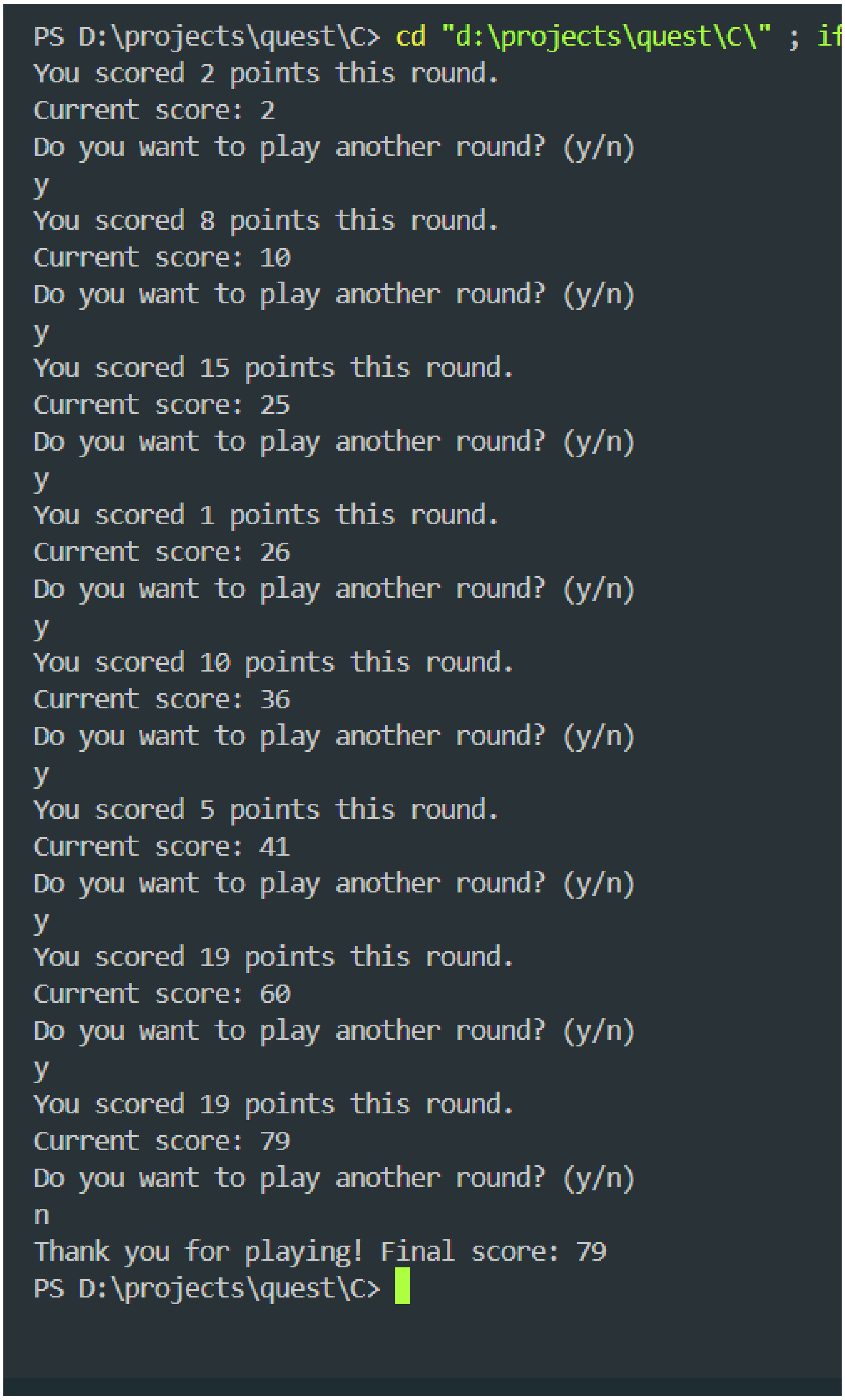
```
volatile variables to simulate clock ticks.
A static variable to count the total number of ticks.
Nested for loops for hours, minutes, and seconds.
if statements to reset counters at appropriate limits.
Key Concepts Covered: Storage classes (volatile, static), Decision-making
(if), Looping (nested for).
 */
#include <stdio.h>
#include <unistd.h>
volatile int tick = 0;
static int total_ticks = 0;
void Tick()
{
    tick = 1;
}
int main() {
    int hours, minutes, seconds;
    for (hours = 0; hours < 24; hours++) {
        for (minutes = 0; minutes < 60; minutes++) {
            for (seconds = 0; seconds < 60; seconds++) {
                Tick();
                if (tick) {
                    tick = 0;
                    total_ticks++;
                    printf("%02d:%02d:%02d\n", hours, minutes, seconds);
                    sleep(1);
                }
            }
        }
    }
    printf("Total ticks: %d\n", total_ticks);
    return 0;
}
```



```
PS D:\projects\quest\C> cc
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
PS D:\projects\quest\C>
```



```c
/*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>
#include <stdlib.h>
void playRound(int* s)
{
    int p = rand() % 20 + 1;
    *s += p;
    printf("You scored %d points this round.\n",p);
}
void main()
{
    static int c_s = 0;
    const int w_s = 100;
    char c;
    while (1) {
        playRound(&c_s);

        if (c_s >= w_s)
        {
            printf("Congratulations! You've won the game with a score of
%d!\n",c_s);
            break;
        }
        else
         {
            printf("Current score: %d\n", c_s);
            printf("Do you want to play another round? (y/n)\n");
            scanf(" %c", &c);
            if (c == 'n' || c == 'N')
            {
                printf("Thank you for playing! Final score: %d\n",c_s);
                break;
            }
        }
    }
}
```



```
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ; if
You scored 2 points this round.
Current score: 2
Do you want to play another round? (y/n)
y
You scored 8 points this round.
Current score: 10
Do you want to play another round? (y/n)
y
You scored 15 points this round.
Current score: 25
Do you want to play another round? (y/n)
y
You scored 1 points this round.
Current score: 26
Do you want to play another round? (y/n)
y
You scored 10 points this round.
Current score: 36
Do you want to play another round? (y/n)
y
You scored 5 points this round.
Current score: 41
Do you want to play another round? (y/n)
y
You scored 19 points this round.
Current score: 60
Do you want to play another round? (y/n)
y
You scored 19 points this round.
Current score: 79
Do you want to play another round? (y/n)
n
Thank you for playing! Final score: 79
PS D:\projects\quest\C>
```