```c
/*Inventory Update System
Input: An array of integers representing inventory levels and an array of changes in stock.
Process: Pass the arrays to a function by reference to update inventory levels.
Output: Print the updated inventory levels and flag items below the restocking threshold.
Concepts: Arrays, functions, pass by reference, decision-making (if-else).*/
#include<stdio.h>
void func(int *stock,int *change)
{
    for(int i=0;i<5;i++)
    {
        stock[i]-=change[i];
    }
    printf("Updated stock\n");
    for(int i=0;i<5;i++)
    printf("For stock %d is %d\n",i+1,stock[i]);
       for(int i=0;i<5;i++)
       {
        if(stock[i]<10)
        printf("Stock %d is below restocking threshold\n",i+1);
       }
}
void main()
{
    int stock[5]={5,10,15,20,25};
    int change[5]={5,6,7,8,11};
    int *s=stock;
    int *c=change;
    func(s,c);
}
```

```
PS D:\projects\quest\C> cd "d:\projects\ques
Updated stock
For stock 1 is 0
For stock 2 is 4
For stock 3 is 8
For stock 4 is 12
For stock 5 is 14
Stock 1 is below restocking threshold
Stock 2 is below restocking threshold
Stock 3 is below restocking threshold
PS D:\projects\quest\C>
```

```c
/*Product Price Adjustment
Input: An array of demand levels (constant) and an array of product
prices.
Process: Use a function to calculate new prices based on demand levels.
The function should return a pointer to an array of adjusted prices.
Output: Display the original and adjusted prices.
Concepts: Passing constant data, functions, pointers, arrays.*/
#include<stdio.h>
#include<stdlib.h>
int * func(int *demand,int *price,int size)
{

    int *ap=(int *)malloc(sizeof(int));
    for(int i=0;i<5;i++)
    {
        if(demand[i]>=3)
        ap[i]=price[i]+demand[i]*3;
        else
        ap[i]=price[i]-demand[i]*3;
    }
    return ap;
}
void main()
{
    int demand[5]={5,4,3,2,1};
    int price[5]={10,20,30,40,50};
```

```
    int adjusted_price[5];
    int *d=demand;
    int *p=price;
    for(int i=0;i<5;i++)
    printf("Stock of %d is %d\n",i+1,p[i]);
    int *ap = func(demand,price,5);
    printf("Adjusted price\n");
    for(int i=0;i<5;i++)
    printf("Stock of %d is %d\n",i+1,ap[i]);
}
```

```
PS D:\projects\quest\C> cd "d:\projects
Stock of 1 is 10
Stock of 2 is 20
Stock of 3 is 30
Stock of 4 is 40
Stock of 5 is 50
Adjusted price
Stock of 1 is 25
Stock of 2 is 32
Stock of 3 is 39
Stock of 4 is 34
Stock of 5 is 47
PS D:\projects\quest\C>
```

```
/*

Daily Sales Tracker
Input: Array of daily sales amounts.
Process: Use do-while to validate sales data input. Use a function to
calculate total sales using pointers.
Output: Display total sales for the day.
Concepts: Loops, arrays, pointers, functions.
*/
#include<stdio.h>
void func(int *sales)
{
    int total=0;
 for(int i=0;i<5;i++)
 {
```

```
    total+= *(sales+i);
 }
 printf("The total sales is %d\n",total);
}
void main()
{
    int sales[5];
    int i=0;
    int*s=sales;
    printf("Enter the daily sales\n");
    do
    {
        scanf("%d",&sales[i]);
        if(sales[i]>0)
        i++;
    } while(i<5);
    func(s);
}
```

```
PS D:\projects\quest\C> cd "d:\
Enter the daily sales
50 60 34 80 90
The total sales is 314
PS D:\projects\quest\C>
```

```
/*Discount Decision System
Input: Array of sales volumes.
Process: Pass the sales volume array by reference to a function. Use a
switch statement to assign discount rates.
Output: Print discount rates for each product.
Concepts: Decision-making (switch), arrays, pass by reference,
functions.*/
#include<stdio.h>
void func(int *volumes)
{
    for(int i=0;i<5;i++)
    {
        switch(*(volumes+i))
        {
```

```
            case 100:printf("Discounted price is %d\n",*(volumes+i)-50);
            break;
            case 200:printf("Discounted price is %d\n",*(volumes+i)-100);
            break;
            case 300:printf("Discounted price is %d\n",*(volumes+i)-150);
            break;
            case 400:printf("Discounted price is %d\n",*(volumes+i)-200);
            break;
            case 500:printf("Discounted price is %d\n",*(volumes+i)-250);
            break;
        }
    }
}
void main()
{
    int volumes[5]={100,200,300,400,500};
    int *v=volumes;
    func(v);
}
```

```
PS D:\projects\quest\C> cd "d:\projec
Discounted price is 50
Discounted price is 100
Discounted price is 150
Discounted price is 200
Discounted price is 250
PS D:\projects\quest\C>
```

```
/*Transaction Anomaly Detector
Input: Array of transaction amounts.
Process: Use pointers to traverse the array. Classify transactions as
"Normal" or "Suspicious" based on thresholds using if-else.
Output: Print classification for each transaction.
Concepts: Arrays, pointers, loops, decision-making.*/
#include<stdio.h>
void func(int *transaction)
{
    for(int i=0;i<5;i++)
```

```
    {
        if(*(transaction+i)>10000)
        printf("Trnsaction id  no: %d of amount %d is
Suspicious\n",i+1,*(transaction+i));
        else
        printf("Trnsaction id  no: %d of amount %d
isNormal\n",i+1,*(transaction+i));
    }
}
void main()
{
    int transaction[5]={1000,2000,15000,12000,7000};
    int *t=transaction;
    func(t);
}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ; if ($?) { g
Trnsaction id  no: 1 of amount 1000 isNormal
Trnsaction id  no: 2 of amount 2000 isNormal
Trnsaction id  no: 3 of amount 15000 is Suspicious
Trnsaction id  no: 4 of amount 12000 is Suspicious
Trnsaction id  no: 5 of amount 7000 isNormal
PS D:\projects\quest\C>
```

```
/*Account Balance Operations
Input: Array of account balances.
Process: Pass the balances array to a function that calculates interest.
Return a pointer to the updated balances array.
Output: Display updated balances.
Concepts: Functions, arrays, pointers, loops.*/
#include<stdio.h>
void func(int *balance)
{
    int r=5,t=4,interest;
    for(int i=0;i<5;i++)
    {
        interest=(*(balance+i)*r*t)/100;
        printf("Interest for %d is %d\n",*(balance+i),interest);
```

```
    }
}
void main()
{
    int balance[5]={1000,1500,4000,2500,7800};
    int *b=balance;
    func(b);
}
```

```
PS D:\projects\quest\C> cd "d:\projects\que
Interest for 1000 is 200
Interest for 1500 is 300
Interest for 4000 is 800
Interest for 2500 is 500
Interest for 7800 is 1560
PS D:\projects\quest\C>
```

```
/*Bank Statement Generator
Input: Array of transaction types (e.g., 1 for Deposit, 2 for Withdrawal)
and amounts.
Process: Use a switch statement to classify transactions. Pass the array
as a constant parameter to a function.
Output: Summarize total deposits and withdrawals.
Concepts: Decision-making, passing constant data, arrays, functions.
*/
#include<stdio.h>
void func(int *transaction,int size)
{
    int deposit=0,withdrawal=0;
    for(int i=0;i<size;i++)
    {
        if(*(transaction+i*2)==1)
        deposit+=*(transaction+i*2+1);
        else
        withdrawal+=*(transaction+i*2+1);
    }
    printf("Total withdrawal is %d\n",withdrawal);
```

```
    printf("Total deposit is %d\n",deposit);

}
void main()
{
    int transaction[10]={1,100,2,200,1,400,2,500,1,600};
    int*t=transaction;
    int size=5;
    func(t,size);
}
```

```
PS D:\projects\quest\C> cd "d:\projects
Total withdrawal is 700
Total deposit is 1100
PS D:\projects\quest\C>
```

```
/*
Loan Eligibility Check
Input: Array of customer credit scores.
Process: Use if-else to check eligibility criteria. Use pointers to update
eligibility status.
Output: Print customer eligibility statuses.
Concepts: Decision-making, arrays, pointers, functions*/
#include<stdio.h>
void func(int *eligibility)
{
    for(int i=0;i<5;i++)
    {
        if(*(eligibility+i)>=5)
        printf("The customer %d is eligible for loan\n",i+1);
        else
        printf("The customer %d is not eligible for loan\n",i+1);
    }
}
void update(int *eligibility,int i,int s)
{
    *(eligibility+i)=s;
```

```
    for(int i=0;i<5;i++)
    {
        printf("%d ",eligibility[i]);
    }
}
void main()
{
    int eligibility[5]={1,2,5,8,9};
    int *e=eligibility;
    func(e);
    update(e,1,6);
}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\"
The customer 1 is not eligible for loan
The customer 2 is not eligible for loan
The customer 3 is eligible for loan
The customer 4 is eligible for loan
The customer 5 is eligible for loan
1 6 5 8 9
PS D:\projects\quest\C>
```

```
/*Order Total Calculator
Input: Array of item prices.
Process: Pass the array to a function. Use pointers to calculate the total
cost.
Output: Display the total order value.
Concepts: Arrays, pointers, functions, loops.*/
#include<stdio.h>
void func(int *price)
{
    int sum=0;
    for(int i=0;i<5;i++)
    {
        sum+=*(price+i);
    }
```

```
    printf("Total cost is %d",sum);
}
void main()
{
    int price[5]={10,20,30,40,50};
    int *p =price;
    func(p);
}
```

```
PS D:\projects\quest\C> cd "d:\projects\ques
Total cost is 150
PS D:\projects\quest\C>
```

```
/*Stock Replenishment Alert
Input: Array of inventory levels.
Process: Use a function to flag products below a threshold. Return a
pointer to flagged indices.
Output: Display flagged product indices.
Concepts: Arrays, functions returning pointers, loops.*/
#include<stdio.h>
void func(int *inventory)
{
    for(int i=0;i<5;i++)
    {
        if(*(inventory+i)<=25)
        printf("Stock no %d is below threshold\n",i+1);
    }
}
void main()
{
    int inventory[5]={10,20,30,40,50};
    int *i=inventory;
    func(i);
}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ; if ($?) {
Stock no 1 is below threshold
Stock no 2 is below threshold
PS D:\projects\quest\C>
```

```c
/*Customer Reward Points
Input: Array of customer purchase amounts.
Process: Pass the purchase array by reference to a function that
calculates reward points using if-else.
Output: Display reward points for each customer.
Concepts: Arrays, functions, pass by reference, decision-making*/
#include<stdio.h>
void func(int *purchase)
{
    for(int i=0;i<5;i++)
    printf("Reward points for %d is %d\n",i+1,*(purchase+i)/10);
}
void main()
{
    int purchase[5]={100,400,200,300,600};
    int *p=purchase;
    func(p);
}
```

```
PS D:\projects\quest\C> cd "d:\proje
Reward points for 1 is 10
Reward points for 2 is 40
Reward points for 3 is 20
Reward points for 4 is 30
Reward points for 5 is 60
PS D:\projects\quest\C>
```

```c
/*Shipping Cost Estimator
Input: Array of order weights and shipping zones.
Process: Use a switch statement to calculate shipping costs based on
zones. Pass the weight array as a constant parameter.
Output: Print the shipping cost for each order.
Concepts: Decision-making, passing constant data, arrays, functions.*/
#include<stdio.h>
void func(int *weight,int const* zone)
{
    for(int i=0;i<5;i++)
    {
        switch(*(zone+i))
        {
            case 1:printf("Shipping cost is %d\n",*(weight+i)*10);
            break;
            case 2:printf("Shipping cost is %d\n",*(weight+i)*20);
            break;
            case 3:printf("Shipping cost is %d\n",*(weight+i)*30);
            break;
            case 4:printf("Shipping cost is %d\n",*(weight+i)*40);
            break;
            case 5:printf("Shipping cost is %d\n",*(weight+i)*50);
            break;
        }
    }
}
```

```c
void main()
{
    int weight[5]={4,5,6,7,8};
    int zone[5]={1,2,3,4,5};
    int *w=weight;
    int *z=zone;
    func(w,z);
}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\
Shipping cost is 40
Shipping cost is 100
Shipping cost is 180
Shipping cost is 280
Shipping cost is 400
PS D:\projects\quest\C>
```

```c
/*Missile Trajectory Analysis
Input: Array of trajectory data points.
Process: Use functions to find maximum and minimum altitudes. Use pointers
to access data.
Output: Display maximum and minimum altitudes.
Concepts: Arrays, pointers, functions.*/
#include<stdio.h>
void func(int * trajectory,int size)
{
    int max=0,min=100;
    for(int i=0;i<size;i++)
    {
        if(*(trajectory+i*2+1)>max)
        max=*(trajectory+i*2+1);
        if(*(trajectory+i*2+1)<min)
        min=*(trajectory+i*2+1);
    }
    printf("The maximum altitude is %d\n",max);
    printf("The minimum altitude is %d\n",min);
}
void main()
```

```
{
    int trajectory[10]={1,1,2,2,3,3,4,5,6,3};
    int *t=trajectory;
    int size=5;
    func(t,size);
}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ; if ($?) { gcc t
The maximum altitude is 5
The minimum altitude is 1
PS D:\projects\quest\C>
```

```
/*Target Identification System
Input: Array of radar signal intensities.
Process: Classify signals into categories using a switch statement. Return
a pointer to the array of classifications.
Output: Display classified signal types.
Concepts: Decision-making, functions returning pointers, arrays.
*/
#include<stdio.h>
#include<stdlib.h>
char * func(int *signal,int size)
{
    char *class=(char *)malloc(size * sizeof(size));
    for(int i=0;i<size;i++)
    {
        switch(*(signal+i))
        {
            case 1:*(class+i)='I';
            break;
            case 2:*(class+i)='A';
            break;
            case 3:*(class+i)='P';
            break;
            case 4:*(class+i)='C';
            break;
            case 5:*(class+i)='S';
            break;
```

```
        }
    }
    return class;
}
void main()
{
    int signal[5]={1,2,3,4,5};
    int *s=signal;
    char *c=func(s,5);
    for(int i=0;i<5;i++)
    {
        switch(*(c+i))
        {
            case 'I':printf("Indian vehicle\n");
            break;
            case 'A':printf("American vehicle\n");
            break;
            case 'P':printf("Persian vehicle\n");
            break;
            case 'C':printf("Chinese vehicle\n");
            break;
            case 'S':printf("Srilankan vehicle\n");
            break;

        }
    }
}
```

```
PS D:\projects\quest\C> cd "
Indian vehicle
American vehicle
Persian vehicle
Chinese vehicle
Srilankan vehicle
PS D:\projects\quest\C>
```

```c
/*
Threat Level Assessment
Input: Array of sensor readings.
Process: Pass the array by reference to a function that uses if-else to
categorize threats.
Output: Display categorized threat levels.
Concepts: Arrays, functions, pass by reference, decision-making.
*/
#include<stdio.h>
void func(int *readings)
{
    for(int i=0;i<5;i++)
    {
        if(*(readings+i)<100)
        printf("Threat level of %d is GREEN\n",*(readings+i));
        else if(*(readings+i)<200)
        printf("Threat level of %d is YELLOW\n",*(readings+i));
        else if(*(readings+i)<300)
        printf("Threat level of %d is ORANGE\n",*(readings+i));
        else if(*(readings+i)<400)
        printf("Threat level of %d is RED\n",*(readings+i));
        else
        printf("Threat level of %d is BLACK\n",*(readings+i));
    }
}
void main()
{
```

```
    int readings[5]={123,78,456,368,299};
    int *r=readings;
    func(r);
}
```

```
PS D:\projects\quest\C> cd "d:\projects
Threat level of 123 is YELLOW
Threat level of 78 is GREEN
Threat level of 456 is BLACK
Threat level of 368 is RED
Threat level of 299 is ORANGE
PS D:\projects\quest\C>
```

```
/*Signal Calibration
Input: Array of raw signal data.
Process: Use a function to adjust signal values by reference. Use pointers
for data traversal.
Output: Print calibrated signal values.
Concepts: Arrays, pointers, functions, loops.*/
#include<stdio.h>
void func(int *signal)
{
    for(int i=0;i<5;i++)
    {
        if(*(signal+i)>6)
        *(signal+i)-=1;
        else if(*(signal+i)<6)
        *(signal+i)+=1;
    }
    printf("Calibrated values\n");
    for(int i=0;i<5;i++)
    printf("%d ",*(signal +i));
}
void main()
{
    int signal[5]={3,4,7,8,5};
```

```
    int *s=signal;
     printf("Original values\n");
    for(int i=0;i<5;i++)
    printf("%d ",*(signal +i));
    printf("\n");
    func(s);
}
```

```
PS D:\projects\quest\C> cd "d:\pr
Original values
3 4 7 8 5
Calibrated values
4 5 6 7 6
PS D:\projects\quest\C>
```

```
/*Matrix Row Sum
Input: 2D array representing a matrix.
Process: Write a function that calculates the sum of each row. The
function returns a pointer to an array of row sums.
Output: Display the row sums.
Concepts: Arrays, functions returning pointers, loops.*/
#include <stdio.h>
#include <stdlib.h>
int* calculate_row_sums(const int *matrix, int rows, int cols) {
    int *row_sums = (int *)malloc(rows * sizeof(int));
    if (row_sums == NULL) {
        printf("Memory allocation failed.\n");
        return NULL;
    }
    for (int i = 0; i < rows; i++) {
        row_sums[i] = 0;
        for (int j = 0; j < cols; j++) {
            row_sums[i] += *(matrix + i * cols + j);
        }
    }
    return row_sums;
}
```

```c
void display_row_sums(const int *row_sums, int rows) {
    printf("Row sums:\n");
    for (int i = 0; i < rows; i++) {
        printf("Row %d: %d\n", i + 1, row_sums[i]);
    }
}
int main() {
    int rows = 3, cols = 4;
    int matrix[3][4] = {
        {1, 2, 3, 4},
        {5, 6, 7, 8},
        {9, 10, 11, 12}
    };
    int *row_sums = calculate_row_sums((int *)matrix, rows, cols);
    if (row_sums == NULL) {
        return 1;
    }
    display_row_sums(row_sums, rows);
    free(row_sums);

    return 0;
}
```

```
PS D:\projects\quest\C> cd "d:\projec
Row sums:
Row 1: 10
Row 2: 26
Row 3: 42
PS D:\projects\quest\C>
```

```c
/*Statistical Mean Calculator
Input: Array of data points.
Process: Pass the data array as a constant parameter. Use pointers to
calculate the mean.
Output: Print the mean value.
Concepts: Passing constant data, pointers, functions.*/
```

```c
#include<stdio.h>
void func(int *data,int size)
{
    int sum=0;
    for(int i=0;i<size;i++)
    sum+=*(data+i);
    printf("Mean is %d",sum/5);
}
void main()
{
    int data[5]={6,7,8,9,10};
    int size =5;
    int *d=data;
    func(d,size);
}
```

```
PS D:\projects\quest\C> cd "d:\projects\qu
Mean is 8
PS D:\projects\quest\C>
```

```c
/*
Temperature Gradient Analysis
Input: Array of temperature readings.
Process: Compute the gradient using a function that returns a pointer to
the array of gradients.
Output: Display temperature gradients.
Concepts: Arrays, functions returning pointers, loops.*/
#include<stdio.h>
#include<stdlib.h>
void func(int *temp,int size)
{
    int *gradient=(int *)malloc((size-1)*sizeof(int));
    for(int i=0;i<size-1;i++)
    {
        gradient[i]=temp[i+1]-temp[i];
        printf("Gradient is %d\n",gradient[i]);
    }
}
```

```c
void main()
{
    int temp[5]={25,23,33,27,37};
    int *t=temp;
    int size=5;
    func(t,size);
}
```

```
PS D:\projects\quest\C> cd "d:\projec
Gradient is -2
Gradient is 10
Gradient is -6
Gradient is 10
PS D:\projects\quest\C>
```

```c
/*Data Normalization
Input: Array of data points.
Process: Pass the array by reference to a function that normalizes values
to a range of 0-1 using pointers.
Output: Display normalized values.
Concepts: Arrays, pointers, pass by reference, functions.*/
#include<stdio.h>
void func(int *data,int size)
{
    for(int i=0;i<size;i++)
    {
       if(*(data+i)>5)
       *(data+i)=1;
       else
       *(data+i)=0;
    }
    printf("Normalized data is\n");
    for(int i=0;i<size;i++)
    printf("%d ",*(data+i));
}
void main()
```

```
{
    int data[5]={2,6,4,8,15};
    int *d=data;
    int size =5;
    func(d,size);
}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ; i
Normalized data is
0 1 0 1 1
PS D:\projects\quest\C>
```

```
/*Exam Score Analysis
Input: Array of student scores.
Process: Write a function that returns a pointer to the highest score. Use
loops to calculate the average score.
Output: Display the highest and average scores.
Concepts: Arrays, functions returning pointers, loops.*/
#include<stdio.h>
int * func(int *scores,int size)
{
    int sum=0,max=0;
    int *h;
    for(int i=0;i<size;i++)
    sum+=scores[i];
    printf("Average score is %d\n",sum/size);
    for(int i=0;i<size;i++)
    {
        if(max<scores[i])
        max=scores[i];
    }
    for(int i=0;i<size;i++)
    {
        if(max==scores[i])
        {
```

```
            h=&scores[i];
            return h;
        }
    }
}
void main()
{
    int scores[5]={57,60,78,98,34};
    int *s=scores;
    int *h;
    int size=5;
    h=func(s,size);
    printf("Highest score is %d",*h);
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINA

PS D:\projects\quest\C> cd "d:\projects\q
Average score is 65
Highest score is 98
PS D:\projects\quest\C>
```

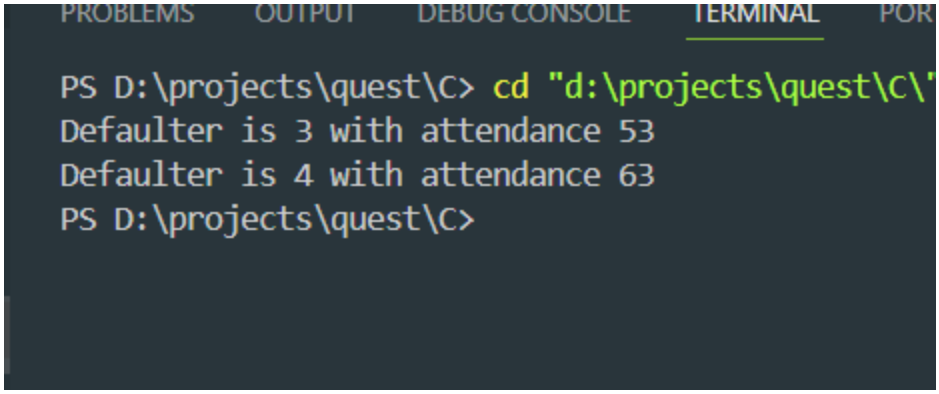
```
/*Grade Assignment
Input: Array of student marks.
Process: Pass the marks array by reference to a function. Use a switch
statement to assign grades.
Output: Display grades for each student.
Concepts: Arrays, decision-making, pass by reference, functions.*/
#include<stdio.h>
void func(int *grades,int size)
{
    for(int i=0;i<size;i++)
    {
        switch(*(grades+i))
        {
            case 90:printf("The grade for student id %d is A\n",i);
            break;
```

```
            case 80:printf("The grade for student id %d is B\n",i);
            break;
            case 70:printf("The grade for student id %d is C\n",i);
            break;
            case 60:printf("The grade for student id %d is D\n",i);
            break;
            case 50:printf("The grade for student id %d is F\n",i);
            break;


        }
    }
}
void main()
{
    int grades[5]={90,80,70,60,50};
    int *g=grades;
    int size=5;
    func(g,size);
}
```

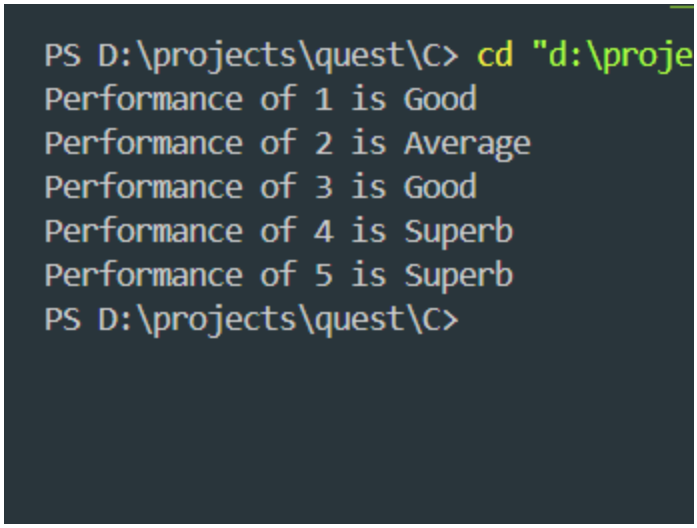
```
PS D:\projects\quest\C> cd "d:\projects\
The grade for student id 0 is A
The grade for student id 1 is B
The grade for student id 2 is C
The grade for student id 3 is D
The grade for student id 4 is F
PS D:\projects\quest\C>
```

```
/*
Student Attendance Tracker
Input: Array of attendance percentages.
Process: Use pointers to traverse the array. Return a pointer to an array
of defaulters.
Output: Display defaulters' indices.
Concepts: Arrays, pointers, functions returning pointers.
*/
#include<stdio.h>
#include<stdlib.h>
```

```c
int count=0;
int *func(int *attendance,int size)
{
    int j=0;
    for(int i=0;i<size;i++)
    {
        if(*(attendance+i)<70)
        count++;
    }
    int *defaulters =(int *)malloc(count*sizeof(int));
    for(int i=0;i<size;i++)
    {
        if(*(attendance+i)<70)
        {
            *(defaulters+j)=*(attendance+i);
            j++;
        }
    }
    return defaulters;
}
void main()
{
    int attendance[5]={77,89,96,53,63};
    int *a=attendance;
    int size=5;
    int *d=func(a,size);
    for(int i=0;i<size;i++)
    {
        for(int j=0;j<count;j++)
        {
            if(*(attendance+i)==*(d+j))
            {
                printf("Defaulter is %d with attendance %d\n",i,*(a+i));
            }
        }
    }


}
```




```c
/*Quiz Performance Analyzer
Input: Array of quiz scores.
Process: Pass the array as a constant parameter to a function that uses
if-else for performance categorization.
Output: Print categorized performance.
Concepts: Arrays, passing constant data, functions, decision-making.*/
#include<stdio.h>
void func(int *scores,int size)
{
    for(int i=0;i<size;i++)
    {
        if(*(scores+i)>=9)
        printf("Performance of %d is Superb\n",i+1);
        else if(*(scores+i)>=7)
        printf("Performance of %d is Good\n",i+1);
         else if(*(scores+i)>=5)
        printf("Performance of %d is Average\n",i+1);
    }
}
void main()
{
    int scores[5]={7,5,8,9,10};
    int *s=scores;
    int size=5;
    func(s,size);
}
```


PS D:\projects\quest\C> cd "d:\proje
Performance of 1 is Good
Performance of 2 is Average
Performance of 3 is Good
Performance of 4 is Superb
Performance of 5 is Superb
PS D:\projects\quest\C>