```c
/*Write a C program that declares an integer pointer, initializes it to
point to an integer variable,
 and prints the value of the variable using the pointer.*/
 #include<stdio.h>
 void main()
 {
    int a=10;
    int *ptr =&a;
    printf("%d",*ptr);
 }
```

```
PS D:\projects\quest\C> cd "d:\proj
10
PS D:\projects\quest\C>
```

```c
/*Write a program that compares two pointers pointing to different
variables of the same type.
 Use relational operators to determine if one pointer points to an address
greater than or less than another and print the results.*/
 #include<stdio.h>
 void main()
 {
    int b,a;
    int *ptr1=&a;
    int *ptr2=&b;
    if(ptr1>ptr2)
    printf("The address of a is greater than b");
    else
    printf("The address of b is greater than a");
 }
```

```
PS D:\projects\quest\C> cd "d:\projects\qu
The address of b is greater than a
PS D:\projects\quest\C>
```

```c
/*Create a program where you declare a pointer to a float variable, assign
a value to the variable,
```

```c
 and then use the pointer to change the value of the float variable.
Print both the original and modified values.*/
#include<stdio.h>
void main()
{
    float a=3.14;
    float *ptr=&a;
    printf("%f\n",*ptr);
    *ptr=*ptr+10;
    printf("%f\n",*ptr);

}
```

```
PS D:\projects\quest\C> cd "d:\proje
3.140000
13.140000
PS D:\projects\quest\C>
```

```c
/*Given an array of integers, write a function that takes a pointer to the
array and its size as arguments.
Use pointer arithmetic to calculate and return the sum of all elements in
the array.*/
#include<stdio.h>
void sum(int *a,int n)
{
    int i=0,sum=0;
    int *ptr;
    for(ptr =a;ptr<a+n;ptr++)
    {
        sum+=*ptr;
    }
    printf("%d",sum);

}
void main()
{
    int a[5]={1,2,3,4,5};
```

```
    sum(a,5);
}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ; if ($
15
PS D:\projects\quest\C>
```

```
/*Write a program that demonstrates the use of a null pointer.
Declare a pointer, assign it a null value, and check if it is null before
attempting to dereference it.*/
#include<stdio.h>
void main()
{
    int *ptr=NULL;
    if(ptr==NULL)
    printf("The pointer is null\n");
    else printf("The pointer is not null");
}
```

```
PS D:\projects\quest\C> cd "d:\projec
The pointer is null
PS D:\projects\quest\C>
```

```
/*Create an example that illustrates what happens when you attempt to
dereference a wild pointer (a pointer that has not been initialized).
Document the output and explain why this leads to undefined behavior.*/
#include<stdio.h>
void main()
{
    int *ptr;
    printf("%d",*ptr);
}
//assigns the pointer to a random memmory
```

```
PS D:\projects\quest\C> cd "d:\projects
17744
PS D:\projects\quest\C>
```

```c
/*Implement a C program that uses a pointer to a pointer. Initialize an
integer variable,
create a pointer that points to it, and then create another pointer that
points to the first pointer.
Print the value using both levels of indirection.*/
#include<stdio.h>
void main()
{
    int n;
    int *ptr1=&n;
    int **ptr2=&ptr1;
    printf("Address of n is %p\n",ptr1);
    printf("Address of ptr1 is %p",ptr2);
}
```

```
PS D:\projects\quest\C> cd "d:\p
tempCodeRunnerFile.c: In functio
tempCodeRunnerFile.c:9:15: warni
     int *ptr2=&ptr1;
               ^
Address of n is 0061FF18
Address of ptr1 is 0061FF14
PS D:\projects\quest\C>
```

```c
/*Write a program that dynamically allocates memory for an array of
integers using malloc.
Populate the array with values, print them using pointers, and then free
the allocated memory.*/
#include<stdio.h>
#include<stdlib.h>
```

```c
void main()
{
    int *a =(int *)malloc(5*sizeof(int));
    printf("Enter elements\n");
    for(int i=0;i<5;i++)
    scanf("%d",&a[i]);
    for(int i=0;i<5;i++)
    printf("%d ",*(a+i));
    free(a);

}
```

```
PS D:\projects\quest\C> cd "d:\pro
Enter elements
1 2 3 4 5
1 2 3 4 5
PS D:\projects\quest\C>
```

```c
/*Define a function that takes two integers as parameters and returns
their sum.
Then, create a function pointer that points to this function and use it to
call the function with different integer values.*/
#include<stdio.h>
int sum(int a,int b)
{
    int s=a+b;
    printf("Sum is %d ",s);
}
void main()
{
    int (*ptr)(int,int)=sum;
    ptr(5,10);
}
```

```
PS D:\projects\quest\C> cd "d:\projects\q
Sum is 15
PS D:\projects\quest\C>
```

```c
/*Create two examples: one demonstrating a constant pointer (where you
cannot change what it points to)
and another demonstrating a pointer to constant data (where you cannot
change the data being pointed to).
Document your findings.*/
#include<stdio.h>
void main()
{
    int a=10,b=5;
    int *const ptr1=&a;
    int const *ptr2=&b;
    printf("Value pointed by ptr1 is %d\n",*ptr1);
    printf("Value pointed by ptr2 is %d\n",*ptr2);
    //*ptr2=*ptr2+5;
   // ptr1 = &b;
}
//error: assignment of read-only location '*ptr2'
 //    *ptr2=*ptr2+5;
 //error: assignment of read-only variable 'ptr1'
  //   ptr1 = &b;
```

```
PS D:\projects\quest\C> cd "d:\proje
Value pointed by ptr1 is 10
Value pointed by ptr2 is 5
PS D:\projects\quest\C>
```

```c
/*Write a program that declares a constant pointer to an integer.
Initialize it with the address of an integer
variable and demonstrate that you can change the value of the integer but
cannot reassign the pointer to point to another variable*/
#include<stdio.h>
void main()
{
    int a=10,b=20;
    int *const ptr=&a;
    printf("Value of *ptr is %d\n",*ptr);
    *ptr=*ptr+10;
    printf("Value of *ptr is %d\n",*ptr);
```

```
    *ptr =&b;

}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest
224.c: In function 'main':
224.c:11:10: warning: assignment makes intege
     *ptr =&b;
          ^
Value of *ptr is 10
Value of *ptr is 20
PS D:\projects\quest\C>
```

```
/*Create a program that defines a pointer to a constant integer.
Attempt to modify the value pointed to by this pointer and observe the
compiler's response.*/
#include<stdio.h>
void main()
{
    int a=20;
    int const * ptr=&a;
    printf("value of *ptr is %d",*ptr);
    *ptr=*ptr+10;

}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ; if ($?) { gcc tempCodeRunn
tempCodeRunnerFile.c: In function 'main':
tempCodeRunnerFile.c:9:9: error: assignment of read-only location '*ptr'
     *ptr=*ptr+10;
         ^
PS D:\projects\quest\C>
```

```
/*Implement a program that declares a constant pointer to a constant
integer.
Show that neither the address stored in the pointer nor the value it
points to can be changed.*/
```

```c
#include<stdio.h>
void main()
{
    int a=10;
    int const * const ptr=&a;
    int b=20;
    *ptr=*ptr+20;
    *ptr=&b;
}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ; if ($?) { gcc
226.c: In function 'main':
226.c:9:9: error: assignment of read-only location '*ptr'
     *ptr=*ptr+20;
         ^
226.c:10:9: error: assignment of read-only location '*ptr'
     *ptr=&b;
         ^
PS D:\projects\quest\C>
```

```c
/*Develop a program that uses a constant pointer to iterate over multiple
integers stored in separate variables.
Show how you can modify their values through dereferencing while keeping
the pointer itself constant.*/
#include <stdio.h>
int main() {
    int arr[4] = {10, 20, 30, 40};
    int *const ptr = arr;
    int i = 0;
    while (i < 4) {
        printf("Value at %d is %d\n", i, *(ptr + i));
        i++;
    }

    i = 0;
    printf("\nNew values \n");

    while (i < 4) {
        *(ptr + i) = *(ptr + i) + 10;
        printf("New value at %d is %d\n", i, *(ptr + i));
```

```
        i++;
    }

    return 0;
}
```

```
PS D:\projects\quest\C> cd "d:\project
Value at 0 is 10
Value at 1 is 20
Value at 2 is 30
Value at 3 is 40

New values
New value at 0 is 20
New value at 1 is 30
New value at 2 is 40
New value at 3 is 50
PS D:\projects\quest\C>
```

```
/*Implement a program that uses pointers and decision-making statements to
check if two constant
integers are equal or not, printing an appropriate message based on the
comparison.*/
#include<stdio.h>
void main()
{
    int a=10,b=10;
    int const *ptr1=&a;
    int const *ptr2=&b;
    if(*ptr1==*ptr2)
    printf("They are equal\n");
    else
    printf("They are not equal\n");
}
```

```
PS D:\projects\quest\C> cd "d:\
They are equal
PS D:\projects\quest\C>
```

```c
/*Create a program that uses conditional statements to determine if a
constant pointer is pointing to a specific value,
printing messages based on whether it matches or not.*/
#include<stdio.h>
void main()
{
    int a=20,b;
    int *const ptr=&a;
    printf("Enter the value\n");
    scanf("%d",&b);
    if(*ptr==b)
    printf("The pointer points to a specified value\n");
    else
    printf("The pointer does not point to a specified value\n");
}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ; if
Enter the value
20
The pointer points to a specified value
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ; if
Enter the value
10
The pointer does not point to a specified value
PS D:\projects\quest\C>
```

```c
/*Write a program that declares two constant pointers pointing to
different integer variables.
Compare their addresses using relational operators and print whether one
points to a higher or lower address than the other.*/
#include<stdio.h>
void main()
```

```
{
    int a,b;
    int const *ptr1=&a;
    int const *ptr2=&b;
    if(ptr1>ptr2)
    printf("Address of a is greater than address of b\n");
    else
    printf("Address of b is greater than address of a\n");
}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ; if ($?
Address of a is greater than address of b
PS D:\projects\quest\C>
```

```
/*Implement a program that uses a constant pointer within
 loops to iterate through multiple variables (not stored in arrays) and
print their values.*/
 #include<stdio.h>
 void main()
 {
    int a=10,b=20,c=30;
    int const*ptr=&a;
    int i=0;
    while(i<=3)
    {
        if(*(ptr+i)==a||*(ptr+i)==b||*(ptr+i)==c)
        printf("The value of ptr is %d\n",*(ptr+i));
        i++;
    }
 }
```

```
 PS D:\projects\quest\C> cd "d:\projects\
 The value of ptr is 10
 The value of ptr is 30
 The value of ptr is 20
 PS D:\projects\quest\C>
```

```c
/*Develop a program that uses a constant pointer to iterate over several
integer
variables (not in an array) using pointer arithmetic while keeping the
pointer itself constant.*/
#include<stdio.h>
void main()
{
    int a=20,b=30,c=40;
    int *const ptr=&a;
    printf("value of a is %d\n",*ptr);
     printf("value of b is %d\n",*(ptr+3));
      printf("value of c is %d\n",*(ptr+2));
}
```

```
PS D:\projects\quest\C> cd "d:\projec
value of a is 20
value of b is 30
value of c is 40
PS D:\projects\quest\C>
```

```c
/*Input: Machine's input power and output power as floats.
Output: Efficiency as a float.
Function: Accepts pointers to input power and output power, calculates
efficiency, and updates the result via a pointer.
Constraints: Efficiency = (Output Power / Input Power) * 100.*/
#include<stdio.h>
float calc(float *p,float *q)
{
    float e;
    e=(*q / *p)*100;
    return e;
}
void main()
{
    float input,output,efficiency;
    float *p1=&input;
    float *p2=&output;
```

```
    printf("Enter the input and output power\n");
    scanf("%f %f",&input,&output);
    efficiency=calc(p1,p2);
    printf("Efficiency is %f",efficiency);
}
```

```
PS D:\projects\quest\C> cd "d:\p
Enter the input and output power
1200 200
Efficiency is 16.666666
PS D:\projects\quest\C>
```

```
/*Input: Current speed (float) and adjustment value (float).
Output: Updated speed.
Function: Uses pointers to adjust the speed dynamically.
Constraints: Ensure speed remains within the allowable range (0 to 100
units).*/
#include<stdio.h>
void func(float *a,float *b)
{
     if((*a + *b)>=0 && (*a + *b)<=100)
     {
        printf("The speed is %f",*a +*b);
     }
     else
     printf("The speed does not remain within allowable range\n");
}
void main()
{
    float speed,adjust;
    float *p=&speed;
    float *q=&adjust;
    printf("Enter the speed and adjustment value\n");
    scanf("%f %f",&speed,&adjust);
    func(p,q);
}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest
Enter the speed and adjustment value
50 20
The speed is 70.000000
PS D:\projects\quest\C>
```

```c
/*Input: Current inventory levels of raw materials (array of integers).
Output: Updated inventory levels.
Function: Accepts a pointer to the inventory array and modifies values
based on production or consumption.
Constraints: No inventory level should drop below zero.*/
#include<stdio.h>
void stock(int *p)
{
    for(int i=0;i<5;i++)
    {
        if(*(p+i)!=0)
        *(p+i)=*(p+i)-10;
    }
    printf("New stock is\n");
    for(int i=0;i<5;i++)
    {
        printf("%d ",*(p+i));
    }
}
void main()
{
    int ar[5];
    int *ptr=ar;
    printf("Enter the inventory levels\n");
    for(int i=0;i<5;i++)
    scanf("%d",&ar[i]);
    stock(ptr);
}
```

```
PS D:\projects\quest\C> cd "d:\projects\
Enter the inventory levels
10 20 30 40 50
New stock is
0 10 20 30 40
PS D:\projects\quest\C>
```

```c
/*Input: Current x, y, z coordinates (integers) and movement delta values.
Output: Updated coordinates.
Function: Takes pointers to x, y, z and updates them based on delta
values.
Constraints: Validate that the coordinates stay within the workspace
boundaries.*/
#include<stdio.h>
void cord(int *x,int *y,int *z,int dx,int dy,int dz)
{
    if((*x+dx)>=0 && (*x+dx)<=10 && (*y+dy)>=0 && (*y+dy)<=10 &&
(*z+dz)>=0 && (*z+dz)<=10)
    {
        *x=*x+dx;
        *y=*y+dy;
        *z=*z+dz;
        printf("Updated cordinates are x : %d ,y : %d ,z : %d",*x,*y,*z);
    }
    else
    printf("The coordinates donot stay within the workspace boundaries");
}
void main()
{
    int x,y,z;
    int *px=&x;
    int *py=&y;
    int *pz=&z;
    int dx,dy,dz;
    printf("Enter the starting cordinate and delta values\n");
    scanf("%d %d %d %d %d %d",&x,&y,&z,&dx,&dy,&dz);
```

```
    cord(px,py,pz,dx,dy,dz);
}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ; if ($
Enter the starting cordinate and delta values
1 1 1 4 4 4
Updated cordinates are x : 5 ,y : 5 ,z : 5
PS D:\projects\quest\C>
```

```
/*Input: Current temperature (float) and desired range.
Output: Adjusted temperature.
Function: Uses pointers to adjust temperature within the range.
Constraints: Temperature adjustments must not exceed safety limits.
 */
#include<stdio.h>
void adjust(float *t,float *r)
{
    float d =30;
    if((*t+d)<= *r)
    printf("adjusted temperature is %f\n",*t+30);
    else
    printf("The temperature exceeds safety limits\n");
}
void main()
{
    float temp,range;
    float *p1 =&temp;
    float *p2=&range;
    printf("Enter the temperature and range\n");
    scanf("%f %f",p1,p2);
    adjust(p1,p2);
}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ; if
Enter the temperature and range
20 100
adjusted temperature is 50.000000
PS D:\projects\quest\C>
```

```c
/*Input: Current tool usage hours (integer) and maximum life span.
Output: Updated remaining life (integer).
Function: Updates remaining life using pointers.
Constraints: Remaining life cannot go below zero.*/
#include<stdio.h>
void func(int *x,int *y)
{
    int rl;
    rl=*y-*x;
    if(rl>=0)
    printf("The remaining life is %d\n",rl);
    else
    printf("Remaining life cannot be less than 0\n");
}
void main()
{
    int uh,ml;
    int *p1=&uh,*p2=&ml;
    printf("Enter the usage hours and max life\n");
    scanf("%d %d",p1,p2);
    func(p1,p2);
}
```

```
PS D:\projects\quest\C> cd "d:\proj
Enter the usage hours and max life
40 100
The remaining life is 60
PS D:\projects\quest\C>
```

```c
/*Input: Weights of materials (array of floats).
```

```c
Output: Total weight (float).
Function: Accepts a pointer to the array and calculates the sum of
weights.
Constraints: Ensure no negative weights are input.*/
#include<stdio.h>
void func(float *p)
{
    float sum=0;
    for(int i=0;i<5;i++)
    {
        sum+=*(p+i);
    }

    printf("The total weight is %f",sum);
}
void main()
{
    float ar[5],w;
    float *ptr=ar;
    printf("Enter the weight of materials\n");
    for(int i=0;i<5;i++)
    {
        do{
            scanf("%f",&w);
          }while(w<0);
          *(ptr+i)=w;
    }
    func(ptr);
}
```

```
PS D:\projects\quest\C> cd "d:\projects\
Enter the weight of materials
10 20 30 40 50
The total weight is 150.000000
PS D:\projects\quest\C>
```

```c
/*
Input: Voltage (float) and current (float).
Output: Updated machine configuration.
```

```c
/*Function: Accepts pointers to voltage and current and modifies their
values.
Constraints: Validate that voltage and current stay within specified
operating ranges.
*/
#include<stdio.h>
void mach(float *p,float *q)
{
    float a=40,b=30;
    if(*p+a<=100 && *q+b<=100)
    printf("Update voltage is %f and current is %f",*p+a,*q+b);
    else
    printf("The voltage and current will exceed safety limits\n");
}
void main()
{
    float v,c;
    float *p1=&v,*p2=&c;
    printf("Enter the voltage and current\n");
    scanf("%f %f",p1,p2);
    mach(p1,p2);
}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ; if ($?) { g
Enter the voltage and current
20 90
The voltage and current will exceed safety limits
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ; if ($?) { g
Enter the voltage and current
29 30
Update voltage is 69.000000 and current is 60.000000
PS D:\projects\quest\C>
```

```c
/*Input: Total products and defective products (integers).
Output: Defect rate (float).
Function: Uses pointers to calculate defect rate = (Defective / Total) *
100.
```

```c
Constraints: Ensure total products > defective products.*/
#include<stdio.h>
void stock(int *x,int *y)
{
    int r;
    if(*x>*y)
    {
        r=((*y*100)/ *x);
        printf("Defective rate is %d",r);

    }
    else
    printf("Number of defective products cannot be greater than total
products");
}
void main()
{
    int t,d;
    int *p1 =&t,*p2=&d;
    printf("Enter the total number of products and defective ones\n");
    scanf("%d %d",p1,p2);
    stock(p1,p2);
}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ; if ($
Enter the total number of products and defective ones
1200 450
Defective rate is 37
PS D:\projects\quest\C>
```

```c
/*Input: Timing intervals between stations (array of floats).
Output: Adjusted timing intervals.
Function: Modifies the array values using pointers.
Constraints: Timing intervals must remain positive.*/
#include<stdio.h>
void mod(float *p)
{
    for(int i=0;i<5;i++)
```

```
    {
        if((*(p+i)-5)>0)
        *(p+i)=*(p+i)-5;
    }
    printf("New timing interval\n");
    for(int i=0;i<5;i++)
    printf("%.2f ",*(p+i));
}
void main()
{
    float a[5];
    float *p=a;
    printf("Enter timing intervals\n");
    for(int i=0;i<5;i++)
    scanf("%f",p+i);
    mod(p);
}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\"
Enter timing intervals
10 20 30 40 50
New timing interval
5.00 15.00 25.00 35.00 45.00
PS D:\projects\quest\C>
```

```
/*Input: Current x, y, z coordinates (floats).
Output: Updated coordinates.
Function: Accepts pointers to x, y, z values and updates them.
Constraints: Ensure updated coordinates remain within machine limits.*/
#include<stdio.h>
void func(float *p,float *q,float *r)
{
    float s=5;
    if(*p+s<=10 && *q+s<=10 && *r+s<=10)
    {
        printf("Updated cordinates are x : %.1f, y : %.1f, z :
%.1f",*p+s,*q+s,*r+s);
```

```
    }
    else
    printf("The cordinates exceed the limits\n");
}
void main()
{
    float x,y,z;
    float *p1=&x,*p2=&y,*p3=&z;
    printf("Enter the coordinates\n");
    scanf("%f %f %f",p1,p2,p3);
    func(p1,p2,p3);
}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ; if ($?) {
Enter the coordinates
2 3 5
Updated cordinates are x : 7.0, y : 8.0, z : 10.0
PS D:\projects\quest\C>
```

```
/*Input: Energy usage data for machines (array of floats).
Output: Total energy consumed (float).
Function: Calculates and updates total energy using pointers.
Constraints: Validate that no energy usage value is negative.*/
#include<stdio.h>
void func(float *a)
{
    float sum=0;
    for(int i=0;i<5;i++)
    {
        sum+=*(a+i);
    }
    printf("Sum is %.1f",sum);
}
void main()
{
    float a[5];
    float *p =a;
```

```
    printf("Enter the energy usage\n");
    for(int i=0;i<5;i++)
    scanf("%f",&a[i]);
    func(p);
}
```

```
PS D:\projects\quest\C> cd "d:\projec
Enter the energy usage
20 10 15 38 40
Sum is 123.0
PS D:\projects\quest\C>
```

```
/*Input: Current production rate (integer) and adjustment factor.
Output: Updated production rate.
Function: Modifies the production rate via a pointer.
Constraints: Production rate must be within permissible limits.*/
#include<stdio.h>
void func(float *p1,float *p2)
{
    if((*p1+*p2)>50)
    printf("production rate not within permissible limits\n");
    else
    printf("Adjusted production rate is %.1f",*p1+*p2);
}
void main()
{
    float pr,af;
    float *p1=&pr,*p2=&af;
    printf("Enter the production rate and adjustment factor\n");
    scanf("%f %f",p1,p2);
    func(p1,p2);
}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C
Enter the production rate and adjustment factor
20 24
Adjusted production rate is 44.0
PS D:\projects\quest\C>
```

```c
/*Input: Quality score (integer) for each product in a batch.
Output: Updated quality metrics.
Function: Updates quality metrics using pointers.
Constraints: Ensure quality scores remain within 0-100.*/
#include<stdio.h>
void func(int *p)
{
    int s=10;
    printf("Updated quality is\n");
    for(int i=0;i<5;i++)
    {
        if((*(p+i)+s)>=0 && (*(p+i)+s)<=100)
        {
            *(p+i)=*(p+i)+s;
            printf("%d ",*(p+i));
        }
    }
}
void main()
{
    int a[5];
    int *p=a;
    printf("Enter the quality\n");
    for(int i=0;i<5;i++)
    scanf("%d",&a[i]);
    func(a);
}
```

```
PS D:\projects\quest\C> cd "d:
Enter the quality
20 30 55 66 17
Updated quality is
30 40 65 76 27
PS D:\projects\quest\C>
```

```c
/*Input: Space used for each section (array of integers).
Output: Updated space allocation.
Function: Adjusts space allocation using pointers.
Constraints: Ensure total space used does not exceed warehouse capacity.*/
#include<stdio.h>
void func(int *a)
{
    int t=30,sum=0;
    for(int i=0;i<5;i++)
    {
        if(t>(sum+*(a+i)+1))
        {
            *(a+i)=*(a+i)+1;
            sum+=*(a+i);
        }
    }

    for(int i=0;i<5;i++)
    printf("%d ",*(a+i));
}
void main()
{
    int a[5];
    int *p=a;
    printf("Enter the spaces\n");
    for(int i=0;i<5;i++)
    scanf("%d",&a[i]);
    func(p);
}
```

```
PS D:\projects\quest\C> cd '
Enter the spaces
2 3 4 5 6
3 4 5 6 7
PS D:\projects\quest\C>
```

```c
/*Input: Machine settings like speed (float) and wrap tension (float).
Output: Updated settings.
Function: Modifies settings via pointers.
Constraints: Validate settings remain within safe operating limits.*/
#include<stdio.h>
void func(float *x,float *y)
{
    float s=10,t=50;
    if(*x+s<100 && *y+t<100)
    {
        *x=*x+s;
        *y=*y+t;
        printf("Updated speed is %f and tension is %f",*x,*y);
    }
    else
    printf("Exceeds safe operating limits\n");
}
void main()
{
    float speed,tension;
    float *p1=&speed;
    float *p2=&tension;
    printf("Enter speed and tension\n");
    scanf("%f %f",p1,p2);
    func(p1,p2);
}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ; if ($?
Enter speed and tension
40 30
Updated speed is 50.000000 and tension is 80.000000
PS D:\projects\quest\C>
```

```c
/*Input: Current temperature (float).
Output: Adjusted temperature.
Function: Adjusts temperature using pointers.
Constraints: Temperature must stay within a specified range.*/
#include<stdio.h>
void func(float *x)
{
    float y=50;
    if(*x+y<100)
    {
        *x=*x+y;
        printf("Temperature is %.1f",*x);
    }
    else
    printf("Temperature must stay within speecifed range\n");
}
void main()
{
    float t;
    float *p=&t;
    printf("Enter the temperature\n");
    scanf("%f",p);
    func(p);
}
```

```
PS D:\projects\quest\C> cd "d
Enter the temperature
30
Temperature is 80.0
PS D:\projects\quest\C>
```

```c
/*Input: Scrap count for different materials (array of integers).
Output: Updated scrap count.
Function: Modifies the scrap count via pointers.
Constraints: Ensure scrap count remains non-negative.*/
#include<stdio.h>
void func(int *x)
{
    int s=5;
    for(int i=0;i<5;i++)
    {
        if(*(x+i)-s>0)
        {
            *(x+i)=*(x+i)-s;
        }

    }
    for(int i=0;i<5;i++)
    {
        printf("%d ",*(x+i));

    }


}
void main()
{
    int a[5];
    int *p=a;
    printf("Enter the count for scraps\n");
    for(int i=0;i<5;i++)
    scanf("%d",&a[i]);
```

```
    func(p);
}
```

```
PS D:\projects\quest\C> cd "d:\proje
Enter the count for scraps
10 20 30 40 50
5 15 25 35 45
PS D:\projects\quest\C>
```

```c
/*Input: Production data for each shift (array of integers).
Output: Updated performance metrics.
Function: Calculates and updates overall performance using pointers.
Constraints: Validate data inputs before calculations.*/
#include<stdio.h>
void func(int *x)
{
    int t=0;
    for(int i=0;i<5;i++)
    {
        if(*(x+i)<0)
        {
            printf("There cannot be a negative perrformance\n");
            break;
        }
        else
        {
            *(x+i)=*(x+i)+10;
            t+=*(x+i);
        }

    }
    printf("Overal performancee is %d",t);
}
void main()
{
    int a[5];
    int *p=a;
    printf("Enter the inputs\n");
    for(int i=0;i<5;i++)
```

```
    scanf("%d",&a[i]);
    func(p);
}
```

```
PS D:\projects\quest\C> cd "d:\p
Enter the inputs
10 20 13 24 15
Overal performancee is 132
PS D:\projects\quest\C>
```