
Practical File

Principles of Programming Language



Principles of Programming Language Lab

Submitted By

Name: Khushi Wadhawan

Roll No.: XXXXXXXXXXXXX

SAP ID: 500093673

B. Tech CSE 1st Sem

School of Computer Science

Submitted to

Mr. Himanshu Sahu

Assistant Professor(S.S.), Cybernetics

School of Computer Science, UPES



School of Computer Science

University of Petroleum and Energy Studies,

Dehradun – 248007: Uttarakhand

Contents

Experiment 1	- 1 -
Experiment 2	- 3 -
Experiment 3	- 11 -
Experiment 4.1	- 21 -
Experiment 4.2	- 27 -
Experiment 5.1	- 34 -
Experiment 5.2	- 44 -
Experiment 6	- 52 -
Experiment 7.1	- 57 -
Experiment 7.2	- 61 -
Experiment 8.1	- 63 -
Experiment 8.2	- 67 -
Experiment 9	- 70 -
Experiment 10	- 76 -

Table of figures:

Output 1	- 5 -
Output 2	- 6 -
Output 3	- 9 -
Output 4	- 10 -
Output 5	- 14 -
Output 6	- 15 -
Output 7	- 17 -
Output 8	- 18 -
Output 9	- 20 -
Output 10	- 22 -
Output 11	- 23 -
Output 12	- 25 -
Output 13	- 26 -
Output 14	- 28 -
Output 15	- 30 -
Output 16	- 32 -
Output 17	- 33 -
Output 18	- 35 -
Output 19	- 37 -
Output 20	- 38 -
Output 21	- 40 -
Output 22	- 43 -
Output 23	- 43 -
Output 24	- 45 -
Output 25	- 46 -
Output 26	- 48 -
Output 27	- 49 -
Output 28	- 51 -
Output 29	- 55 -
Output 30	- 56 -

Output 31	- 58 -
Output 32	- 60 -
Output 33	- 62 -
Output 34	- 64 -
Output 35	- 66 -
Output 36	- 69 -
Output 37	- 71 -
Output 38	- 73 -
Output 39	- 75 -
Output 40	- 78 -

Experiment 1

1. Study of Linux Commands

a. Working with Directories: *mkdir, rmdir, dir, pwd, cd, ls*

mkdir – Make a directory, this command creates a new directory with the name path.

rmdir – Remove a directory, this command removes a directory whose name is given by path.

dir – This command lists all the files and subdirectories in a specific directory.

pwd – Print working Directory, it prints the path of specified directory from the root.

cd – Change directory, this command changes the current directory to specified directory.

ls – This command lists the current directory's contents.

b. Handling Files: *vi, gedit, more, cp, mv, rm*

vi – Files can be edited using the screen-oriented text editor **vi**. This editor enables you to edit lines in context with other lines in the file.

gedit – It is a powerful general purpose text editor in Linux. It is the default text editor of the GNOME desktop environment.

more – This command is used to view the text files in the command prompt, displaying on screen at a time in case the file is large.

cp – cp stands for copy. This command is used to copy files or group of files or directories.

mv – mv stands for move. This command is used to move one or more files or directories from one place to another.

rm – rm stands for remove. This command is used to remove objects such as files, directories etc.

2. Familiarization of the following

a. Structure of C program [Example: C program to print “My name is ...”]

```
#include<stdio.h>    //header file
int main()           //function
{ // beginning of main() body
    char name[20];    //initializing variable
    printf(“enter name”);
    scanf(“%s”, &name);    //taking input from user
    printf(“My name is %s”, name);    // printing output
    return 0;         //standard for successful execution of the program
} // end of main() body
```

.

Experiment 2

1. Given 2 numbers. Calculate sum, difference, multiplication and division.

Algorithm:

Step 1: Start

Step 2: Declare variables a, b, sum, diff, mul, div.

Step 3: Read variables a, b.

Step 4: $\text{sum} = a + b$

Step 5: $\text{diff} = a - b$

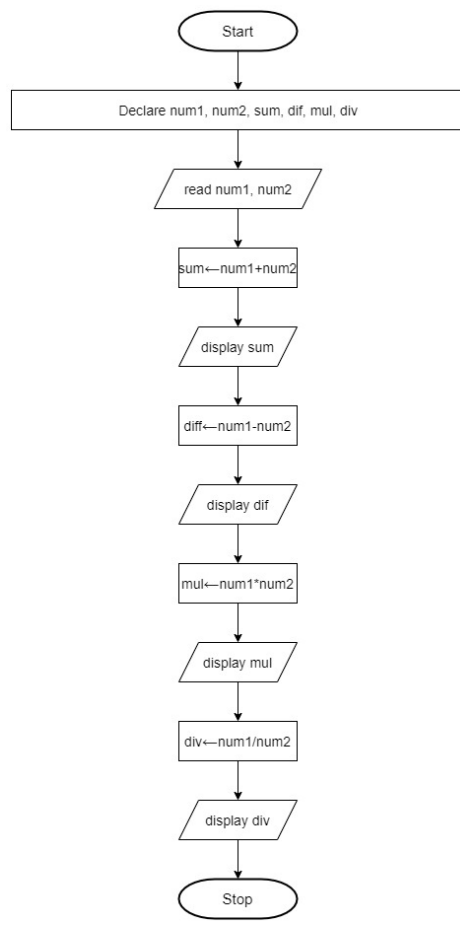
Step 6: $\text{mul} = a * b$

Step 7: $\text{div} = a / b$

Step 8: Display sum, diff, mul, div.

Step 9: Stop

Flowchart:



Code:


```

#include<stdio.h>

int main()
{
    float num1, num2, sum, dif, mul, div;
    printf("Enter the value for the two numbers");
    scanf("%f%f",&num1,&num2);
    sum = num1+num2;
    dif = num1-num2;
    mul = num1*num2;
    div = num1/num2;
    printf("The sum of %fv and %f is %f\n",num1,num2,sum);
    printf("The difference of %f and %f is %f\n",num1,num2,dif);
    printf("The product of %f and %f is %f\n",num1,num2,mul);
    printf("The division of %f and %f is %f\n",num1,num2,div);
    return 0;
}

```

Test cases:

Input	Output			
10 5	15.000000	5.000000	50.000000	2.000000
4 2	6.000000	2.000000	8.000000	2.000000
-6 3	-3.000000	-9.000000	-18.000000	-2.000000
-7 -16	-23.000000	9.000000	112.000000	0.437500
12 60	72.000000	-48.000000	720.000000	0.200000

Output:

```
khushi15@khushi15-VirtualBox:~$ ./a.out
Enter the value for the two numbers10
5
The sum of 10.000000 and 5.000000 is 15.000000
The difference of 10.000000 and 5.000000 is 5.000000
The product of 10.000000 and 5.000000 is 50.000000
The division of 10.000000 and 5.000000 is 2.000000
```

Output 1

2. Find if the given number is even or not.

Algorithm:

Step 1: Start

Step 2: Declare variables a, b.

Step 3: Read variable a.

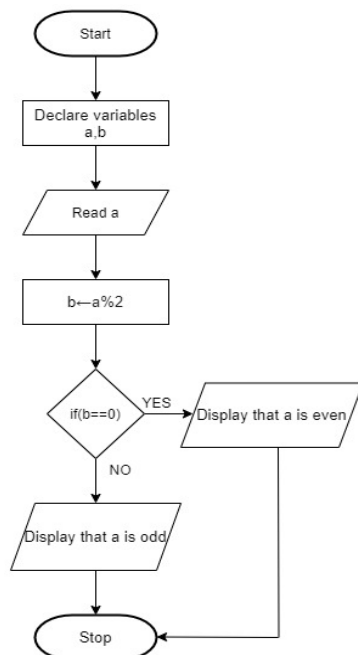
Step 4: $b = a \% 2$

Step 5: Check if b is equal to 0

Step 6: If true then display that a is even.

Step 7: If false then display that a is odd.

Step 8: Stop

Flowchart:

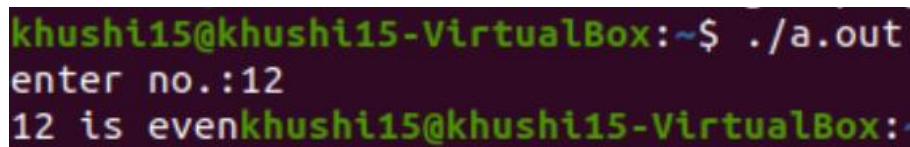
Code:

```
#include<stdio.h>

int main()
{
    int a,b;
    printf("enter no.:");
    scanf("%d",&a);
    b=a%2;
    if(b==0)
        printf("%d is even",a);
    else
        printf("%d is odd",a);
    return 0;
}
```

Test cases:

Input	Output
2	2 is even
5	5 is odd
17	17 is odd
56	56 is even
9	9 is odd

Output:

```
khushi15@khushi15-VirtualBox:~$ ./a.out
enter no.:12
12 is evenkhushi15@khushi15-VirtualBox:~$
```

Output 2

3. Find the biggest of three numbers.

Algorithm:

Step 1: Start

Step 2: Declare variables a, b, c, new.

Step 3: Read variables a, b, c.

Step 4: Check if a is greater than b.

Step 5: If true then new=a.

Step 6: If false then new=b.

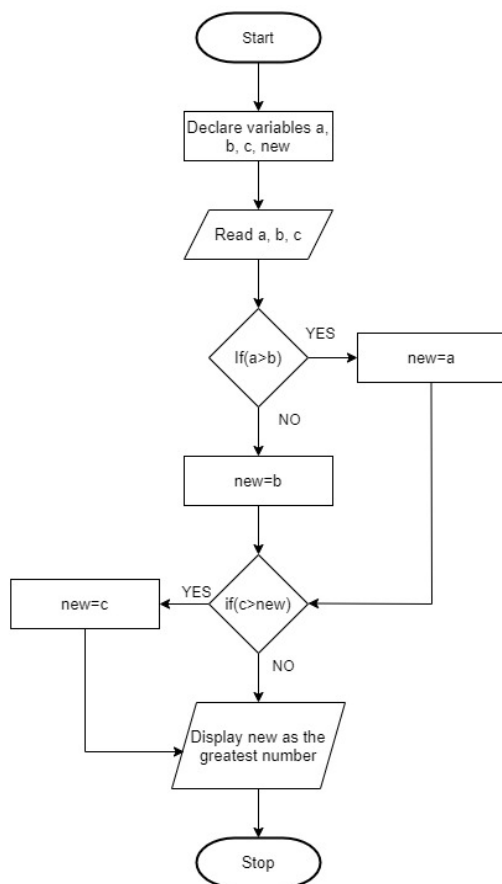
Step 7: Check if c is greater than new.

Step 8: If true then new=c.

Step 9: Display the value of new as the greatest number.

Step 10: Stop

Flowchart:



Code:

```
#include<stdio.h>
int main()
{
    int a,b,c,new;
    printf("enter 3 no.s:");
    scanf("%d%d%d",&a,&b,&c);
    if(a>b)
        new=a;
    if(b>a)
        new=b;
    if(c>new)
        new=c;
    printf("the greatest number out of %d %d and %d
is:%d",a,b,c,new);
    return 0;
}
```

Test cases:

Input	Output
-1 14 5	the greatest number out of -1 14 and 5 is: 14
20 4 9	the greatest number out of 20 4 and 9 is: 20
2 3 8	the greatest number out of 2 3 and 8 is: 8
-2 -4 -6	the greatest number out of -2 -4 and -6 is: -2
32 7 10	the greatest number out of 32 7 and 10 is: 32

Output:

```
khushi15@khushi15-VirtualBox:~$ ./a.out
enter 3 no.s:10
3
24
the greatest number out of 10 3 and 24 is:24
```

Output 3

4. Multiply two numbers without using arithmetic multiplication operator (*).

Algorithm:

Step 1: Start

Step 2: Declare variables a, b, res, new.

Step 3: res=0

Step 4: Read variables a, b.

Step 5: Check if a is less than b.

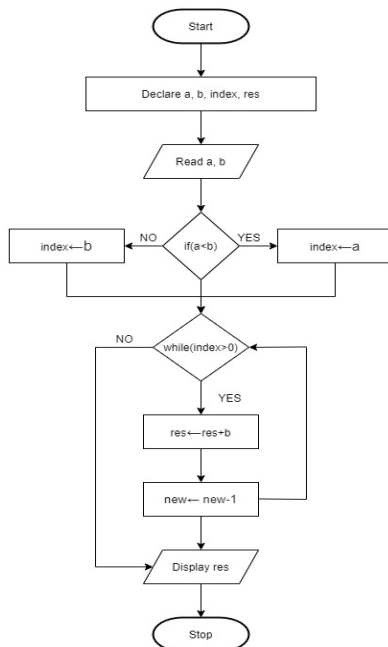
Step 6: If true then new=a.

Step 7: If false then new=b.

Step 8: res=res+b and new=new-1 while new is greater than 0

Step 9: Display res as the product of a and b.

Step 10: Stop.

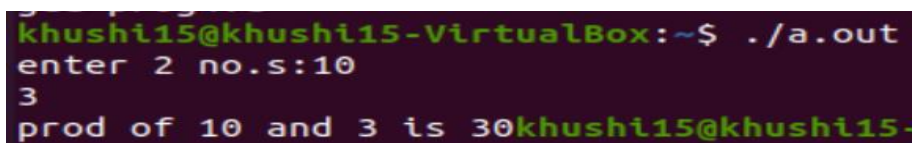
Flowchart:

Code:

```
#include<stdio.h>
int main()
{
    int a,b,res,new;
    res=0;
    printf("enter 2 no.s:");
    scanf("%d %d",&a,&b);
    new=a;
    while(new>0)
    {
        res=res+b;
        new=new-1;
    }
    printf("prod of %d and %d is %d",a,b,res);
    return 0;
}
```

Test Cases:

Input	Output
3 4	12
-9 2	-18
-1 -5	5
0 6	0
8 7	56

Output:

```
khushi15@khushi15-VirtualBox:~$ ./a.out
enter 2 no.s:10
3
prod of 10 and 3 is 30khushi15@khushi15-
```

Output 4

Experiment 3

1. Obtain the required inputs and compute the areas of the following shapes: (i) Parallelogram (with base and height), (ii) Trapezoid (with height, long base, short base), (iii) Rhombus (with height and side), (iv) Sphere (with radius), (v) Ellipse (with major and minor radius)

Algorithm:

Step 1: Start

Step 2: Declare choice, var1, var2, var3, area.

Step 3: Read choice.

Step 4: If choice = 1, read var1, var2, calculate area of parallelogram ($\text{var1} * \text{var2}$) and display value of area.

Step 5: If choice = 2, read var1, var2, var3 calculate area of trapezoid $((\text{var1} + \text{var2}) * \text{var3} / 2)$ and display value of area.

Step 6: If choice = 3, read var1, var2, calculate area of rhombus $((\text{var1} + \text{var2}) / 2)$ and display value of area.

Step 7: If choice = 4, read var1, calculate area of sphere $(4 * 3.14 * \text{var1} * \text{var1})$ and display value of area.

Step 8: If choice = 5, read var1, var2, calculate area of ellipse $(3.14 * \text{var1} * \text{var2})$ and display value of area.

Step 9: If choice = 0, exit the program.

Step 10: Stop.

Code:

```
#include<stdio.h>
int main()
{
    int choice;
    float var1, var2, var3, area;
    while(1)
    {
        printf("\n Enter the shape for which you want to
calculate the area\n");
        printf("\n 1.Parallelogram \t 2. Trapezoid \t 3.
Rhombus \t 4. Sphere \t 5.Ellipse \t 0.for exit");
```



```
scanf("%d", &choice);
switch(choice)
{
case 1:
{
printf("\n Enter the base and height of the
parallelogram\n");
scanf("%f%f",&var1,&var2);
area = var1*var2;
printf("The area of the parallelogram is %f\n",area);
break;
}
case 2:
{
printf("\n Enter the value of parallel side a,b &
height of the trapezoid\n");
scanf("%f%f%f",&var1,&var2, &var3);
area = (var1+var2)*var3/2;
printf("The area of the trapezoid is %f\n",area);
break;
}
case 3:
{
printf("\n Enter the value of diagonals a,b");
scanf("%f%f",&var1,&var2);
area = (var1*var2)/2;
printf("The area of the rhombus is %f\n",area);
break;
}
case 4:
{
```

```
        printf("\n Enter the value of radius");
        scanf("%f",&var1);
        area = 4*3.14*var1*var1;
        printf("The area of the sphere is %f\n",area);
        break;
    }
    case 5:
    {
        printf("\n Enter the value of length of major and minor
axis\n");
        scanf("%f%f",&var1,&var2);
        area = 3.14*var1*var2;
        printf("The area of the ellipse is %f\n",area);
        break;
    }
    case 0:
    {
        return 0;
        break;
    }
    default:
    {
        printf("The choice is incorrect\n");
        break;
    }
    }
    return 0;
}
```

Test Cases:

Input	Output
1	Enter the base and height of parallelogram
5 2	The area of the parallelogram is 10.000000
2	Enter the value of parallel side a,b & height of the trapezoid
3 4 7	The area of the trapezoid is 24.500000
3	Enter the value of parallel diagonals a,b
6 3	The area of the rhombus is 9.000000
4	Enter the value of radius
4	The area of the sphere is 200.960007
5	Enter the value of length of major and minor axis
3 1	The area of the ellipse is 9.420000

Output:

```

khushi15@khushi15-VirtualBox:~$ ./a.out

Enter the shape for which you want to calculate the area
1.Parallelogram      2. Trapezoid      3. Rhombus      4. Sphere      5.Elli
pse      0.for exit1

Enter the base and height of the parallelogram
5
2
The area of the parallelogram is 10.000000

Enter the shape for which you want to calculate the area
1.Parallelogram      2. Trapezoid      3. Rhombus      4. Sphere      5.Elli
pse      0.for exit2

Enter the value of parallel side a,b & height of the trapezoid
3
4
7
The area of the trapezoid is 24.500000

Enter the shape for which you want to calculate the area
1.Parallelogram      2. Trapezoid      3. Rhombus      4. Sphere      5.Elli
pse      0.for exit3

```

Output 5

```

pse      0.for exit3

Enter the value of diagonals a,b6
3
The area of the rhombus is 9.000000

Enter the shape for which you want to calculate the area

1.Parallelogram      2. Trapezoid      3. Rhombus      4. Sphere      5.Elli
pse      0.for exit4

Enter the value of radius4
The area of the sphere is 200.960007

Enter the shape for which you want to calculate the area

1.Parallelogram      2. Trapezoid      3. Rhombus      4. Sphere      5.Elli
pse      0.for exit5

Enter the value of length of major and minor axis
3
1
The area of the ellipse is 9.420000

Enter the shape for which you want to calculate the area

1.Parallelogram      2. Trapezoid      3. Rhombus      4. Sphere      5.Elli
pse      0.for exit0
khushi15@khushi15-VirtualBox:~$

```

Output 6

2. Given two numbers. Demonstrate the swapping of the values (i) using a third variable (ii) without using a third variable.

Algorithm:

Step 1: Start

Step 2: Declare a, b, temp.

Step 3: Read a, b.

Step 4: temp=a

Step 5: a=b

Step 6: b=temp

Step 7: Display values of a and b.

Step 8: Stop.

Code:(using third variable)

```

#include <stdio.h>

int main()
{
    int a,b,temp;
    printf("enter 2 integers to swap:");

```

```
scanf("%d %d",&a,&b);  
temp = a;  
a = b;  
b =temp;  
printf("after swapping, first integer = %d and second  
integer = %d",a,b);  
}
```

Code:(without using third variable)

```
#include <stdio.h>  
int main()  
{  
    int a,b,temp;  
    printf("enter 2 integers to swap:");  
    scanf("%d %d",&a,&b);  
    a=a+b;  
    b=a-b;  
    a=a-b;  
    printf("after swapping, first integer = %d and second  
integer = %d",a,b);  
}
```

Test Cases:

Input	Output
2 4	after swapping, first integer = 4 and second integer = 2
5 0	after swapping, first integer = 0 and second integer = 5
-8 3	after swapping, first integer = 3 and second integer = -8
1 1	after swapping, first integer = 1 and second integer = 1
20 17	after swapping, first integer = 17 and second integer = 20

Output:

```
khushi15@khushi15-VirtualBox:~$ ./a.out
enter 2 integers to swap:10
5
after swapping, first integer = 5 and second integer = 10k
```

Output 7

3. Convert temperature from Celsius to Fahrenheit and Kelvin.**Algorithm:**

Step 1: Start

Step 2: Declare celsius, fahr, kevin.

Step 3: Read celsius.

Step 4: $\text{fahr} = 1.8 * \text{celsius} + 32.0$

Step 5: $\text{kelvin} = \text{celsius} + 273.0$

Step 6: Display fahr and kelvin.

Step 7: Stop.

Code:

```
#include<stdio.h>
int main()
{
    float celsius, fahr, kelvin;
    printf("Enter temperature in celsius: ");
    scanf("%f", &celsius);
    fahr = 1.8 * celsius + 32.0;
    kelvin = 273.15 + celsius;
    printf("%f Celsius = %f Fahrenheit\n", celsius, fahr);
    printf("%f Celsius = %f Kelvin", celsius, kelvin);
    return(0);
}
```

Test Cases:

Input	Output
20	20.000000 Celsius = 68.000000 Fahrenheit 20.000000 Celsius = 293.149994 Kelvin
56	56.000000 Celsius = 132.800003 Fahrenheit 56.000000 Celsius = 329.149994 Kelvin
2	2.000000 Celsius = 35.599998 Fahrenheit 2.000000 Celsius = 275.149994 Kelvin
38	38.000000 Celsius = 100.400002 Fahrenheit 38.000000 Celsius = 311.149994 Kelvin
32	32.000000 Celsius = 89.599998 Fahrenheit 32.000000 Celsius = 305.149994 Kelvin

Output:

```
khushi15@khushi15-VirtualBox:~$ ./a.out
Enter temperature in celsius: 32
32.000000 Celsius = 89.599998 Fahrenheit
32.000000 Celsius = 305.149994 Kelvinkhushi15
```

Output 8

4. Print the given days in years-month-days format. E.g. 396 days = 1 year, 1 month, 1 day

Algorithm:**Step 1:** Start**Step 2:** Declare ndays, y, m, d.**Step 3:** Read ndays.**Step 4:** $y = (\text{int})\text{ndays}/365;$ **Step 5:** $\text{ndays} = \text{ndays} - (365 * y);$ **Step 6:** $m = (\text{int})\text{ndays}/30;$ **Step 7:** $d = (\text{int})\text{ndays} - (m * 30);$ **Step 8:** Display y, m, d.**Step 9:** Stop.

Code:

```
#include<stdio.h>
int main()
{
    int ndays,y,m,d;
    printf("input number of days:");
    scanf("%d",&ndays);
    y=(int)ndays/365;
    ndays=ndays-(365*y);
    m=(int)ndays/30;
    d=(int)ndays-(m*30);
    printf("%d Year(s) \n%d Month(s) \n%d Day(s)",y,m,d);
    return 0;
}
```

Test Case:

Input	Output
400	1 Year(s) 1 Month(s) 5 Day(s)
231	1 Year(s) 1 Month(s) 5 Day(s)
543	1 Year(s) 5 Month(s) 28 Day(s)
98	0 Year(s) 3 Month(s) 8 Day(s)
767	2 Year(s) 1 Month(s)

7 Day(s)

Output:

```
khushi15@khushi15-VirtualBox:~$ ./a.out
input number of days:428
1 Year(s)
2 Month(s)
3 Day(s)
```

Output 9

Experiment 4.1

List of lab activities:

1. Find the biggest of 3 numbers.

Algorithm:

Step 1: Start

Step 2: Declare variables a, b, c, gr.

Step 3: Read variables a, b, c.

Step 4: gr=a.

Step 5: Check if b is greater than gr.

Step 6: If true then gr=b.

Step 7: Check if c is greater than gr.

Step 8: If true then gr=c.

Step 9: Display the value of gr as the biggest number.

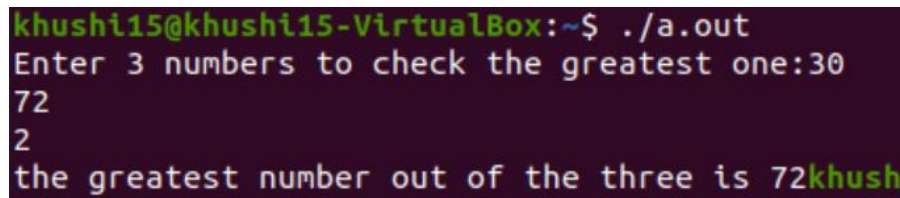
Step 10: Stop.

Code:

```
#include<stdio.h>
int main()
{
    int a,b,c,gr;
    printf("Enter 3 numbers to check the greatest one:");
    scanf("%d%d%d",&a,&b,&c);
    gr=a;
    if(b>gr)
        gr=b;
    if(c>gr)
        gr=c;
    printf("the greatest number out of the three is %d",gr);
    return 0;
}
```

Test Cases:

-1 14 5	the greatest number out of the three is 14
20 4 9	the greatest number out of the three is 20
2 3 8	the greatest number out of the three is 8
-2 -4 -6	the greatest number out of the three is -2
32 7 10	the greatest number out of the three is 32

Output:

```
khushi15@khushi15-VirtualBox:~$ ./a.out
Enter 3 numbers to check the greatest one:30
72
2
the greatest number out of the three is 72khush
```

Output 10

2. Check whether the given year is leap year or not.

Algorithm:

Step 1: Start
Step 2: Declare year.
Step 3: Read year.
Step 4: Check if year is divisible by 400.
Step 5: If true then, it is a leap year.
Step 6: Check if year is divisible by 100.
Step 7: If true then, it is not a leap year.
Step 8: Check if year is divisible by 4.
Step 9: If true then, It is a leap year.
Step 10: Stop.

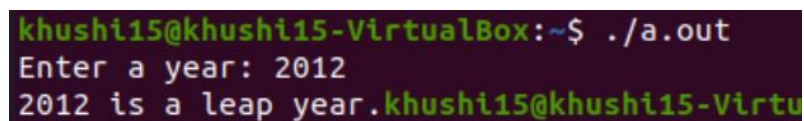
Code:

```
#include <stdio.h>
int main()
{
    int year;
    printf("Enter a year: ");
```

```
scanf("%d", &year);
if (year % 400 == 0)
{
    printf("%d is a leap year.", year);
}
if (year % 100 == 0)
{
    printf("%d is not a leap year.", year);
}
if (year % 4 == 0)
{
    printf("%d is a leap year.", year);
}
else
{
    printf("%d is not a leap year.", year);
}
return 0;
}
```

Test Cases:

Input	Output
2013	2013 is not a leap year.
1990	1990 is not a leap year.
2012	2012 is a leap year.
2061	2060 is a leap year.
1740	1740 is a leap year.

Output:

```
khushi15@khushi15-VirtualBox:~$ ./a.out
Enter a year: 2012
2012 is a leap year.khushi15@khushi15-Virtu
```

Output 11

List of practice activities:

1. Find whether a given number is even or odd.

Algorithm:

Step 1: Start

Step 2: Declare variable a.

Step 3: Read variable a.

Step 4: Check if $a \% 2 == 0$

Step 5: If true then display that a is even.

Step 6: If false then display that a is odd.

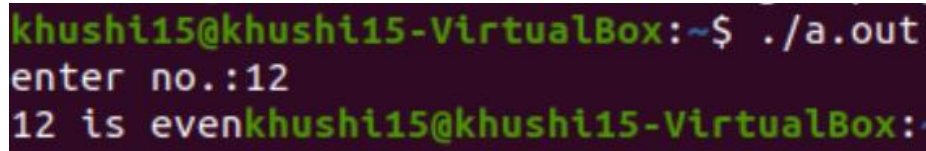
Step 7: Stop

Code:

```
#include<stdio.h>
int main()
{
    int a;
    printf("Enter a number:");
    scanf("%d",&a);
    if(a%2==0)
        printf("\n%d is even",a);
    else
        printf("\n%d is odd",a);
    return 0;
}
```

Test Cases:

Input	Output
2	2 is even
5	5 is odd
17	17 is odd
56	56 is even
9	9 is odd

Output:

```
khushi15@khushi15-VirtualBox:~$ ./a.out
enter no.:12
12 is evenkhushi15@khushi15-VirtualBox:~$
```

Output 12

2. Find if the given number is positive, negative or zero.

Algorithm:

Step 1: Start

Step 2: Declare a.

Step 3: Read a.

Step 4: Check if $a=0$.

Step 5: If true, display that a is 0.

Step 6: Check if $a>0$.

Step 7: If true, display that a is positive.

Step 8: Check if $a<0$.

Step 9: If true, display that a is negative.

Step 10: Stop.

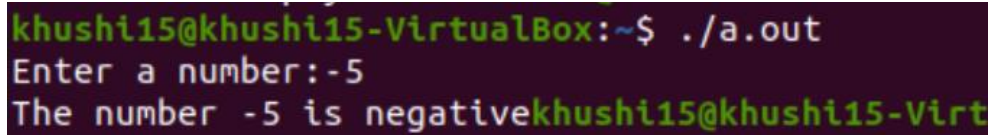
Code:

```
#include<stdio.h>
int main()
{
    int a;
    printf("Enter a number:");
    scanf("%d",&a);
    if(a==0)
        printf("The number %d is zero",a);
    if(a>0)
        printf("The number %d is positive",a);
    if(a<0)
        printf("The number %d is negative",a);
    return 0;
```

}

Test Cases:

Input	Output
0	The number 0 is zero
-7	The number -7 is negative
8	The number 8 is positive
23	The number 23 is positive
-52	The number 52 is negative

Output:

```
khushi15@khushi15-VirtualBox:~$ ./a.out
Enter a number:-5
The number -5 is negativekhushi15@khushi15-Virt
```

Output 13

Experiment 4.2

List of lab activities:

1. Find the roots of a quadratic equation

Algorithm:

Step 1: Start

Step 2: Declare x, y, z, det, root1, root2, real, img.

Step 3: Read x, y, z.

Step 4: $\text{det} = y^2 - 4xz$.

Step 5: If $\text{det} > 0$, $\text{root1} = (-y + \sqrt{\text{det}}) / (2 * x)$, $\text{root2} = (-y - \sqrt{\text{det}}) / (2 * x)$, and display the values of root1 and root2.

Step 6: If $\text{det} = 0$, $\text{root1} = \text{root2} = -y / (2 * x)$, and display the value of root1 and root2.

Step 7: If $\text{det} < 0$, $\text{real} = -y / (2 * x)$, $\text{img} = \sqrt{-\text{det}} / (2 * x)$, and display $\text{root1} = \text{real} + \text{img}$ and $\text{root2} = \text{real} - \text{img}$.

Step 8: Stop.

Code:

```
#include<stdio.h>
#include<math.h>
int main()
{
    float x, y, z, det, root1, root2, real, img;
    printf("\n Enter the value of coefficient x, y and z: \n
");
    scanf("%f %f %f", &x, &y, &z);
    det = y*y-4*x*z;
    if (det > 0)
    {
        root1 = (-y + sqrt(det)) / (2 * x);
        root2 = (-y - sqrt(det)) / (2 * x);
        printf("\n Value of root1 = %.2f and value of root2 =
%.2f", root1, root2);
```



```
}  
else if (det == 0)  
{  
    root1 = root2 = -y / (2 * x); // both roots are equal;  
    printf("\n Value of root1 = %.2f and Value of root2 =  
%.2f", root1, root2);  
}  
else  
{  
    real = -y / (2 * x);  
    img = sqrt(-det) / (2 * x);  
    printf("\n value of root1 = %.2f + %.2fi and value of  
root2 = %.2f - %.2fi ", real, img, real, img);  
}  
}
```

Test Cases:

Input	Output
3 5 7	value of root1= -0.83+1.28i and value of root2 = -0.83-1.28i
2.3 4 5.6	value of root1= -0.87+1.30i and value of root2= -0.87-1.30i
1 1 1	value of root1= -0.50+0.87i and value of root2= -0.50-0.87i

Output:

```
Enter the value of coefficient x, y and z:  
3  
5  
7  
  
value of root1 = -0.83 + 1.28i and value of root2 = -0.83-1.28i
```

Output 14

2. Check if a given character is a vowel or consonant using Switch-Case statement.

Algorithm:

Step 1: Start

Step 2: Declare ch.

Step 3: Read ch.

Step 4: Check if ch is uppercase or lowercase version of the alphabets 'a' 'e' 'i' 'o' 'u'.

Step 5: If true, display that ch is a vowel.

Step 6: If false, display that ch is a consonant.

Step 7: Stop.

Code:

```
#include <stdio.h>
int main()
{
    char ch;
    printf("Enter any alphabet: ");
    scanf("%c", &ch);
    switch(ch)
    {
        case 'a':
        case 'A':
        case 'e':
        case 'E':
        case 'i':
        case 'I':
        case 'o':
        case 'O':
        case 'u':
        case 'U':
            printf("Vowel");
            break;
        default:
            printf("Consonant");
    }
```

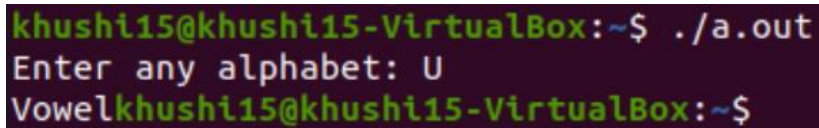
```
}

    return 0;

}
```

Test Cases:

Input	Output
A	Vowel
v	Consonant
B	Consonant
o	Vowel
U	Vowel

Output:

```
khushi15@khushi15-VirtualBox:~$ ./a.out
Enter any alphabet: U
Vowelkhushi15@khushi15-VirtualBox:~$
```

Output 15

List of practice activities:

1. Check if the given number is divisible by 2 and 3 or not.

Algorithm:

Step 1: Start

Step 2: Declare a.

Step 3: Read a.

Step 4: Check if $a\%2=0$ and $a\%3=0$.

Step 5: If true, a is divisible by 2 and 3.

Step 6: If false, is not divisible by 2 and 3 both.

Step 7: Stop.

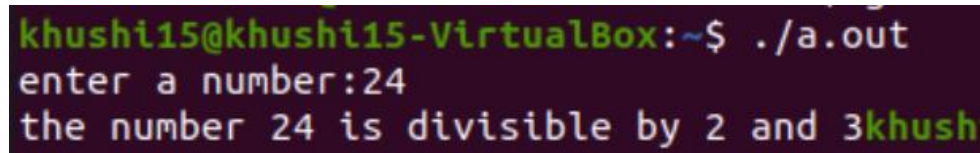
Code:

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

```
int main()
{
    int a;
    printf("enter a number:");
    scanf("%d",&a);
    if(a%2==0 && a%3==0)
        printf("the number %d is divisible by 2 and 3",a);
    if(a%2==0 && a%3!=0)
        printf("the number %d is divisible by 2 but not by
3",a);
    if(a%2!=0 && a%3==0)
        printf("the number %d is divisible by 3 but not by
2",a);
    if(a%2!=0 && a%3!=0)
        printf("the number %d is not divisible by 2 or by
3",a);
    return 0;
}
```

Test Cases:

Input	Output
6	the number 6 is divisible by 2 and 3
9	the number 9 is divisible by 3 but not by 2
4	the number 4 is divisible by 2 but not by 3
7	the number 7 is not divisible by 2 or by 3
28	the number 28 is divisible by 2 but not by 3

Output:

```
khushi15@khushi15-VirtualBox:~$ ./a.out
enter a number:24
the number 24 is divisible by 2 and 3khush
```

Output 16

2. Check whether a given character is an alphabet, digit or special character.

Algorithm:

Step 1: Start

Step 2: Declare ch.

Step 3: Read ch.

Step 4: If ch is between a-z or A-Z, display that it is an alphabet.

Step 5: If ch is between 0-9, display that it is a digit.

Step 6: Else, display that ch is a special character.

Step 7: Stop.

Code:

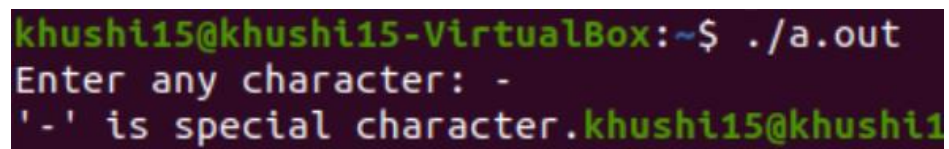
```
#include <stdio.h>
int main()
{
    char ch;
    printf("Enter any character: ");
    scanf("%c", &ch);
    if((ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z'))
    {
        printf("'%c' is alphabet.", ch);
    }
    if(ch >= '0' && ch <= '9')
    {
        printf("'%c' is digit.", ch);
    }
    else
    {

```

```
        printf("'%' is special character.", ch);  
    }  
    return 0;  
}
```

Test Cases:

Input	Output
F	F is alphabet
?	? is special character
8	8 is digit
u	u is alphabet
&	& is special character

Output

```
khushi15@khushi15-VirtualBox:~$ ./a.out  
Enter any character: -  
'-' is special character.khushi15@khushi15
```

Output 17

Experiment 5.1

List of Lab Activities:

1. Given positive number 'n', generate all the Armstrong numbers between 1 and n. [Hint: A 3-digit number (Ex. 153) is an Armstrong number if the sum of cube of each digit ($1^3+5^3+3^3$) is equal to 153]

Algorithm:

Step 1: Start

Step 2: Declare a, b, c, i, sum=0.

Step 3: Read a.

Step 4: Make a loop which runs a number of times.

Step 5: Within that loop, while $c > 0$, $b = c \% 10$, $sum = sum + (b * b * b)$, $c = c / 10$.

Step 6: Check if $sum = i$.

Step 7: If true, Print i as an Armstrong number.

Step 8: Stop.

Code:

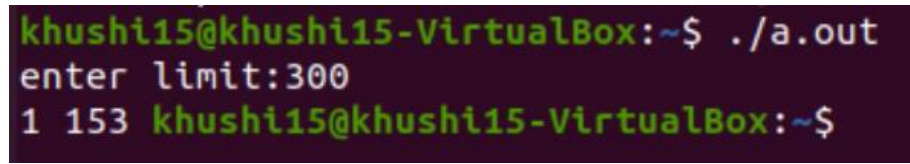
```
#include<stdio.h>

int main()
{
    int a,b,c,i,sum=0;
    printf("enter limit:");
    scanf("%d",&a);
    for(i=1;i<=a;i++)
    {
        c=i;
        sum=0;
        while(c>0)
        {
            b=c%10;
            sum=sum+(b*b*b);
            c=c/10;
        }
    }
}
```

```
        if (sum==i)
            printf("%d ",i);
    }
}
```

Test Cases:

Input	Output
10	1
100	1
200	1 153
300	1 153
500	1 153 370 371 407

Output:

```
khushi15@khushi15-VirtualBox:~$ ./a.out
enter limit:300
1 153 khushi15@khushi15-VirtualBox:~$
```

Output 18

2. *Multiple two given numbers without using the arithmetic binary multiplication operator using for loop.*

Algorithm:

Step 1: Start

Step 2: Declare a, b, i, m=0.

Step 3: Read a, b.

Step 4: Check if a>b, If true, make a for loop that runs b number times and m=m+a.

Step 5: Display m.

Step 6: Check if a<b, If true, make a for loop that runs a number times and m=m+b.

Step 7: Display m.

Step 8: Stop.

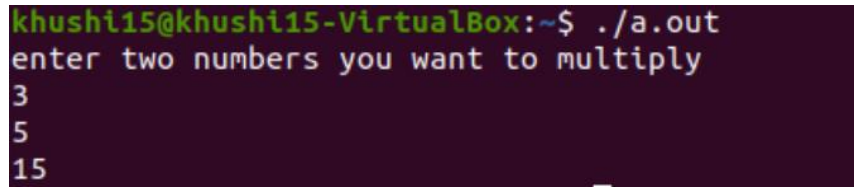
Code:

```
#include<stdio.h>
int main()
{
    int a,b,i,m=0;
    printf("enter two numbers you want to multiply\n");
    scanf("%d%d",&a,&b);
    if(a>b)
    {
        for(i=0;i<b;i++)
        {
            m=m+a;
        }
        printf("%d\n",m);
    }
    else
    {
        for(i=0;i<a;i++)
        {
            m=m+b;
        }
        printf("%d\n",m);
    }
}
```

Test Cases:

Input	Output
3 4	12
-9 2	-18
-1 -5	5
0 6	0

8 7 56

Output:

```
khushi15@khushi15-VirtualBox:~$ ./a.out
enter two numbers you want to multiply
3
5
15
```

*Output 19***List of Practice Activities:**

1. Generate all the prime numbers between 1 and n, where n is a value supplied by the user.

Algorithm:

Step 1: Start

Step 2: Declare i, j, n.

Step 3: Read n.

Step 4: Make a loop that runs n number of times.

Step 5: Withing that loop, make another loop that runs i number of times.

Step 6: Check if $i \% j = 0$, if true, $c++$

Step 7: Outside the loop, check if $c=2$, If true, print c as a prime number.

Step 8: Stop.

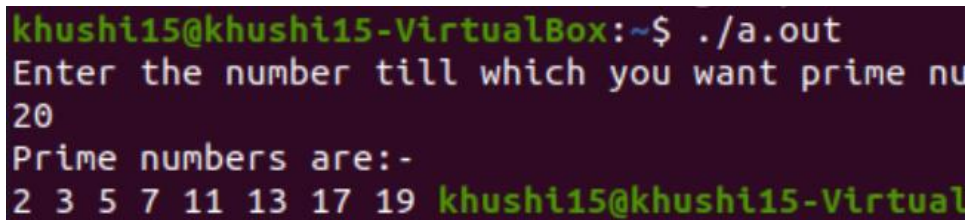
Code:

```
#include<stdio.h>
int main()
{
    int i,j,n;
    printf("Enter the number till which you want prime
numbers\n");
    scanf("%d",&n);
    printf("Prime numbers are:-\n");
    for(i=2;i<=n;i++)
    {
```

```
int c=0;
for(j=1;j<=i;j++)
{
    if(i%j==0)
    {
        c++;
    }
}
if(c==2)
{
    printf("%d ",i);
}
}
```

Test Cases:

Input	Output
5	2 3 5
10	2 3 5 7
15	2 3 5 7 11 13
20	2 3 5 7 9 11 13 17 19
25	2 3 5 7 9 11 13 17 19 23

Output:

```
khushi15@khushi15-VirtualBox:~$ ./a.out
Enter the number till which you want prime nu
20
Prime numbers are:-
2 3 5 7 11 13 17 19 khushi15@khushi15-Virtual
```

Output 20

2. Reverse the digits of a given number and check if given number is a Palindrome or not using do-while.

Algorithm:

Step 1: Start

Step 2: Declare a, b, c=0, d.

Step 3: Read a.

Step 4: d=a.

Step 5: In a do while loop while d>0, b=d%10, c=c*10+b, d=d/10.

Step 6: Check if a=c.

Step 7: If true, display a as a palindrome number.

Step 8: If false, display a is not a palindrome number.

Step 9: Stop.

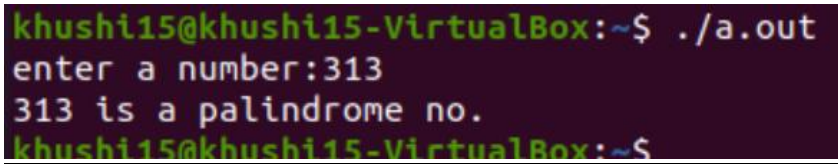
Code:

```
#include<stdio.h>
int main()
{
    int a,b,c=0,d;
    printf("enter a number:");
    scanf ("%d",&a);
    d=a;
    do
    {
        b=d%10;
        c=c*10+b;
        d=d/10;
    }while (d>0);
    if(a==c)
    {
        printf("%d is a palindrome no.\n",a);
    }
    else
    {
```

```
        printf("%d is not a palindrome no.\n",a);  
    }  
    return 0;  
}
```

Test Cases:

Input	Output
123	123 is not a palindrome no.
444	444 is a palindrome no.
30	30 is not a palindrome no.
66	66 is a palindrome no.
909	909 is a palindrome no.

Output:

```
khushi15@khushi15-VirtualBox:~$ ./a.out  
enter a number:313  
313 is a palindrome no.  
khushi15@khushi15-VirtualBox:~$
```

Output 21

3. Using a menu driven control, print the given patterns as per user choice.

Algorithm:

- Step 1:** Start
- Step 2:** Declare a.
- Step 3:** Read a.
- Step 4:** Make an infinite loop and display loop menu in it.
- Step 5:** If a=1, display rectangular number pattern.
- Step 6:** If a=2, display triangular number pattern.
- Step 7:** If a=3, display star pyramid.
- Step 8:** If a=0, exit program
- Step 9:** Stop.

Code:

```
#include <stdio.h>  
  
int main()
```

```
{
    int a;
    while(1)
    {
        printf("Enter 1. For rectangular number pattern\n2. For
triangular number pattern\n3. For star pyramid pattern\n0. To
Exit\n");
        scanf("%d",&a);
        switch(a)
        {
            case 1:
            {
                int i,sum=0;
                for(i=1;i<6;i++)
                {
                    sum=i*10000+i*1000+i*100+i*10+i;
                    printf("%d\n",sum);
                }
                break;
            }
            case 2:
            {
                int i,j;
                for(i=1;i<6;i++)
                {
                    for(j=1;j<=i;j++)
                        printf("%d",i);
                    printf("\n");
                }
                break;
            }
        }
    }
}
```

```
case 3:
{
    int i,j,k;
    for(i=1;i<=4;i++)
    {
        for(j=1;j<=4-i;j++)
            printf(" ");
        for(k=1;k<=2*i-1;k++)
            printf("*");
        printf("\n");
    }
    break;
}
case 0:
{
    return 0;
    break;
}
default:
    printf("wrong entry");
}
}
```

Test Cases:

Input	Output
1	1111 2222 3333 4444 5555

2 1
 22
 333
 4444
 55555

3

```

      *
     ***
    *****
   ********
  
```

Output:

```

khushi15@khushi15-VirtualBox:~$ ./a.out
Enter 1. For rectangular number pattern
2. For triangular number pattern
3. For star pyramid pattern
0. To Exit
1
11111
22222
33333
44444
55555
Enter 1. For rectangular number pattern
2. For triangular number pattern
3. For star pyramid pattern
0. To Exit
2
1
22
333
4444
55555
Enter 1. For rectangular number pattern
2. For triangular number pattern
3. For star pyramid pattern
0. To Exit
3
*

```

Output 22

```

0. To Exit
3
*
***
*****
*****
Enter 1. For rectangular number pattern
2. For triangular number pattern
3. For star pyramid pattern
0. To Exit
0
khushi15@khushi15-VirtualBox:~$

```

Output 23

Experiment 5.2

List of Lab Activities:

1. Find the sum of digits of a number using while loop.

Algorithm:

Step 1: Start

Step 2: Declare a, b, d, sum=0.

Step 3: Read a.

Step 4: d=a.

Step 5: In a loop, while d>0, b=d%10, sum=sum+b, d=d/10.

Step 6: Outside the loop, print sum as sum of digits.

Step 7: Stop.

Code:

```
#include<stdio.h>
int main()
{
    int a,b,d,sum=0;
    printf("enter a number:");
    scanf("%d",&a);
    d=a;
    while (d>0)
    {
        b=d%10;
        sum=sum+b;
        d=d/10;
    }
    printf("\nsum of digits of %d is %d ",a,sum);
}
```

Test Cases:

Input	Output
23	sum of digits of 23 is 5
605	sum of digits of 605 is 11
9	sum of digits of 9 is 9
48	sum of digits of 48 is 12
5421	sum of digits of 5421 is 12

Output:

```
khushi15@khushi15-VirtualBox:~$ ./a.out
enter a number:235
sum of digits of 235 is 10 khushi15@khushi15-
```

Output 24

2. Given value of 'n', find the sum of the series $1 + 1/2 + 1/3 + 1/4 + 1/5 + \dots + 1/n$.

Algorithm:

Step 1: Start

Step 2: Declare n, sum=0.0, i, tn.

Step 3: Read n.

Step 4: Make a loop that runs n number of times.

Step 5: In loop, $tn=1/tn$ and $sum= sum+tn$.

Step 6: Outside loop, print sum.

Step 7: Stop.

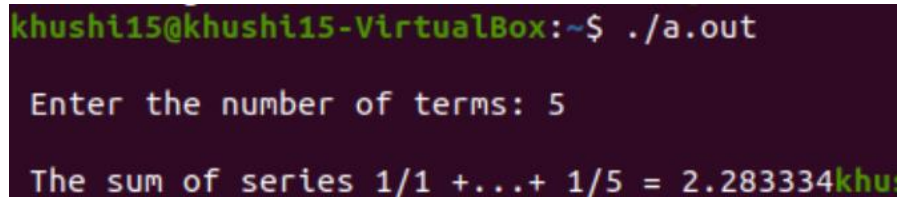
Code:

```
#include <stdio.h>
int main()
{
    int n;
    float sum=0.0,i,tn;
    printf("\n Enter the number of terms: ");
    scanf ("%d",&n);
```

```
for(i=1.0;i<=n;i++) {  
    tn=1/i;  
    sum=sum+tn;  
}  
printf("\n The sum of series 1/1 +...+ 1/%d = %f",n,sum);  
return 0;  
}
```

Test Cases:

Input	Output
2	The sum of series $1/1 + \dots + 1/2 = 1.500000$
3	The sum of series $1/1 + \dots + 1/3 = 1.833333$
5	The sum of series $1/1 + \dots + 1/5 = 2.283334$
7	The sum of series $1/1 + \dots + 1/7 = 2.592857$
10	The sum of series $1/1 + \dots + 1/10 = 2.928968$

Output:

```
khushi15@khushi15-VirtualBox:~$ ./a.out  
Enter the number of terms: 5  
The sum of series 1/1 +...+ 1/5 = 2.283334
```

Output 25

3. Print the given pattern using nested for loop.

Algorithm:**Code:****Test Cases:****Output:**

List of Practice Activities:

1. Given positive number 'n', generate all the Armstrong numbers between 1 and n. [Hint: A 3-digit number (Ex. 153) is an Armstrong number if the sum of cube of each digit ($1^3+5^3+3^3$) is equal to 153]

Algorithm:

Step 1: Start

Step 2: Declare a, b, c, i, sum=0.

Step 3: Read a.

Step 4: Make a loop which runs a number of times.

Step 5: Within that loop, while $c > 0$, $b = c \% 10$, $sum = sum + (b * b * b)$, $c = c / 10$.

Step 6: Check if $sum == i$.

Step 7: If true, Print i as an Armstrong number.

Step 8: Stop.

Code:

```
#include<stdio.h>
int main()
{
    int a,b,c,i,sum=0;
    printf("enter limit:");
    scanf("%d",&a);
    for(i=1;i<=a;i++)
    {
        c=i;
        sum=0;
        while(c>0)
        {
            b=c%10;
            sum=sum+(b*b*b);
            c=c/10;
        }
        if(sum==i)
            printf("%d ",i);
    }
}
```

```
}  
}
```

Test Cases:

Input	Output
10	1
100	1
200	1 153
300	1 153
500	1 153 370 371 407

Output:

```
khushi15@khushi15-VirtualBox:~$ ./a.out  
enter limit:300  
1 153 khushi15@khushi15-VirtualBox:~$
```

Output 26

2. Generate the first 'n' terms of a Fibonacci sequence. [Hint: The first and second terms of a Fibonacci sequence are 0 and 1. Subsequent terms are found by adding the preceding two terms in the sequence.]

Algorithm:

Step 1: Start

Step 2: Declare i, n, t1=0, t2=1, sum=2, nextTerm=t1+t2;

Step 3: Read n.

Step 4: Print t1, t2, and nextTerm.

Step 5: Make a loop that runs n-3 number of times, in the loop, t1 = t2, t2 = nextTerm, nextTerm = t1 + t2 and display nextTerm.

Step 6: Stop.

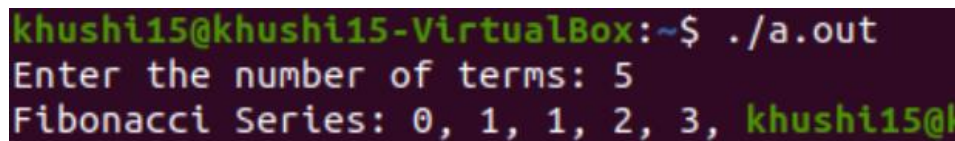
Code:

```
#include <stdio.h>  
  
int main()
```

```
{  
    int i, n;  
    int t1 = 0, t2 = 1, sum=2;  
    int nextTerm = t1+t2;  
    printf("Enter the number of terms: ");  
    scanf("%d", &n);  
    printf("Fibonacci Series: %d, %d, %d, ", t1, t2,  
nextTerm);  
    for (i=3;i<n;++i)  
    {  
        t1 = t2;  
        t2 = nextTerm;  
        nextTerm = t1 + t2;  
        printf("%d, ", nextTerm);  
    }  
    return 0;  
}
```

Test Cases:

Input	Output
3	Fibonacci Series: 0, 1, 1,
4	Fibonacci Series: 0, 1, 1, 2
6	Fibonacci Series: 0, 1, 1, 2, 3, 5,
8	Fibonacci Series: 0, 1, 1, 2, 3, 5, 8, 13,
10	Fibonacci Series: 0, 1, 1, 2, 3, 5, 8, 13, 21, 42

Output:

```
khushi15@khushi15-VirtualBox:~$ ./a.out  
Enter the number of terms: 5  
Fibonacci Series: 0, 1, 1, 2, 3, khushi15@khushi15-VirtualBox:~$
```

Output 27

3. *Print numbers which are divisible by 3 and 5 from the first 'n' natural numbers.*

Algorithm:

Step 1: Start

Step 2: Declare a, i.

Step 3: Read a.

Step 4: Make a loop that runs a number of times.

Step 5: In the loop, check if $i \% 3 = 0$ and $i \% 5 = 0$.

Step 6: If true, print i.

Step 7: Stop.

Code:

```
#include<stdio.h>
int main()
{
    int a,i;
    printf("enter limit:");
    scanf("%d",&a);
    for(i=1;i<=a;i++)
    {
        if(i%3==0 && i%5==0)
        {
            printf("%d ",i);
        }
    }
    return 0;
}
```

Test Cases:

Input	Output
10	
15	15
50	15 30 45
60	15 30 45 60

80

15 30 45 60 75

Output:

```
khushi15@khushi15-VirtualBox:~$ ./a.out  
enter limit:50  
15 30 45 khushi15@khushi15-VirtualBox:~$
```

Output 28

Experiment 6

1. Function `main()` gets a number and calls the following three functions

- a. “`void armstrong(int)`” checks if the given number is an armstrong number or not.
- b. “`void coprime(int)`” reverses the given number and checks if the given number and reversed number are coprime.
- c. “`int factorial(int)`” computes the factorial of the given number using recursion and returns to `main()`.

Algorithm:

Step 1: Start

Step 2: Declare a, b, c, fact, arm, co.

Step 3: Read a.

Step 4: Call function `facto` with parameter a.

Step 5: Read b.

Step 6: Call function `armstrong` with parameter b.

Step 7: Read c.

Step 8: Call function `coprime` with parameter c.

Step 9: Stop

Code:

```
#include<stdio.h>
int facto(int a);
void armstrong(int b);
void coprime(int c);
int main()
{
    int a,fact,b,arm,c,co;
    printf("enter a number:");
    scanf("%d",&a);
    fact=facto(a);
    printf("the factorial of %d is %d",a,fact);
    printf("\nenter a number:");
```

```
scanf ("%d", &b);
armstrong(b);
printf("\nenter a number:");
scanf ("%d", &c);
coprime(c);
return 0;
}
int facto(int a)
{
    int res=1;
    while(a>0)
    {
        res=res*a;
        a--;
    }
    return res;
}
void armstrong(int b)
{
    int a,c,sum=0;
    c=b;
    sum=0;
    while(c>0)
    {
        a=c%10;
        sum=sum+(a*a*a);
        c=c/10;
    }
    if(sum==b)
        printf("%d is an armstrong number ",b);
```

```
        else
            printf("%d is not an armstrong number ",b);
    }
    void coprime(int c)
    {
        int a,i,temp,rev=0,d,e=0;
        d=c;
        while(d>0)
        {
            a=d%10;
            rev=rev*10+a;
            d=d/10;
        }
        if(c>rev)
            temp=rev;
        else
            temp=c;
        for(i=1;i<=temp;i++)
        {
            if(c%i==0 && rev%i==0)
                e++;
        }
        if(e==1)
            printf("%d and %d are coprime numbers",c,rev);
        else
            printf("%d and %d are not coprime numbers",c,rev);
    }
```

Output:

```
enter a number:4
the factorial of 4 is 24
enter a number:567
567 is not an armstrong number
enter a number:76
76 and 67 are coprime numberskhushi15@khus
```

Output 29

2. Function `main()` gets two numbers from the user and calls three functions in the given order:
- `"int triangle_area(int base, int height)"` returns the area of the right-angled triangle to `main()`.
 - `"void swap(int *, int*)"` swaps the two numbers using bitwise operator and displays them.
 - `"float* remainder (int a, int b)"` returns the remainder of `a/b` to `main()`.

Algorithm:

Step 1: Start

Step 2: Declare a, b.

Step 3: Read a,b.

Step 4: Call function `triangle_base` with parameters a,b.

Step 5: Call function `swap` with parameters a,b.

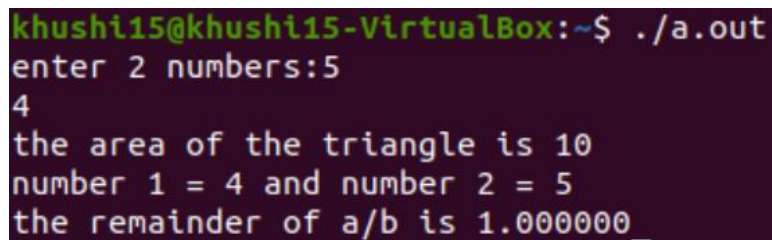
Step 6: Call function `remainder` with parameters a,b.

Step 7: Stop

Code:

```
#include<stdio.h>
int triangle_base(int base,int height);
void swap(int *x,int *y);
float rem(int a,int b);
int main()
{
    int a,b;
    printf("enter 2 numbers:");
    scanf("%d%d",&a,&b);
```

```
    triangle_base(a,b);
    swap(&a,&b);
    printf("number 1 = %d and number 2 = %d\n",a,b);
    rem(a,b);
    return 0;
}
int triangle_base(int base, int height)
{
    int area=0.5*base*height;
    printf("the area of the triangle is %d\n",area);
}
void swap(int *x, int *y)
{
    *x = *x ^ *y;
    *y = *x ^ *y;
    *x = *x ^ *y;
}
float rem(int a,int b)
{
    float r=a%b;
    printf("the remainder of a/b is %f \n",r);
}
```

Output:

```
khushi15@khushi15-VirtualBox:~$ ./a.out
enter 2 numbers:5
4
the area of the triangle is 10
number 1 = 4 and number 2 = 5
the remainder of a/b is 1.000000
```

Output 30

Experiment 7.1

1. Find sum of all array elements using recursion.

Algorithm:

Step 1: Start

Step 2: Define function sum.

Step 3: Declare MAX_SIZE, arr[], N, i, sumofarray.

Step 4: Read N.

Step 5: Call function sum.

Step 6: Stop.

Code:

```
#include <stdio.h>
int sum(int arr[], int start, int len);
int main()
{
    int MAX_SIZE = 100;
    int arr[MAX_SIZE];
    int N, i, sumofarray;
    printf("Enter size of the array: ");
    scanf("%d", &N);
    printf("Enter elements in the array: ");
    for(i=0; i<N; i++)
    {
        scanf("%d", &arr[i]);
    }
    sumofarray = sum(arr, 0, N);
    printf("Sum of array elements: %d", sumofarray);
    return 0;
}
int sum(int arr[], int start, int len)
{
```

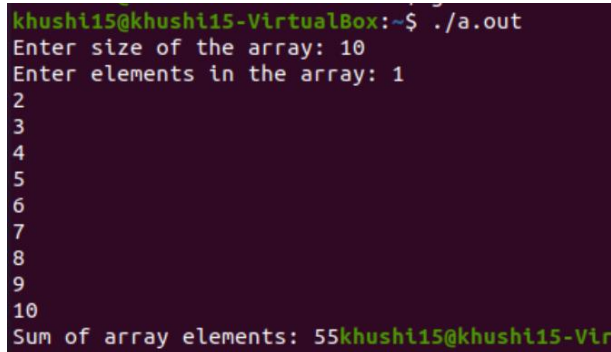
```

    if(start >= len)
        return 0;
    return (arr[start] + sum(arr, start + 1, len));
}

```

Test cases:

Input	Output
0 -	0
3 1 2 3	6
5 0 -9 3 2 7	3
7 -3 -2 -1 -4 -5 -6 -7	-28

Output:


```

khushi15@khushi15-VirtualBox:~$ ./a.out
Enter size of the array: 10
Enter elements in the array: 1
2
3
4
5
6
7
8
9
10
Sum of array elements: 55khushi15@khushi15-Vir

```

Output 31

2. Create an array 'a1' with 'n' elements. Insert an element in ith position of 'a1' and also delete an element from jth position of 'a1'.

Algorithm:**Step 1:** Start**Step 2:** Declare position, i, a[], size**Step 3:** Read size.**Step 4:** Run for loop to accept elements of array from user.**Step 5:** Accept ith element from user**Step 6:** Accept position of element from user and delete it from array using `a[i]=a[i+1]` in for loop from `i=position` to `i=size`.**Step 7:** Display new array.

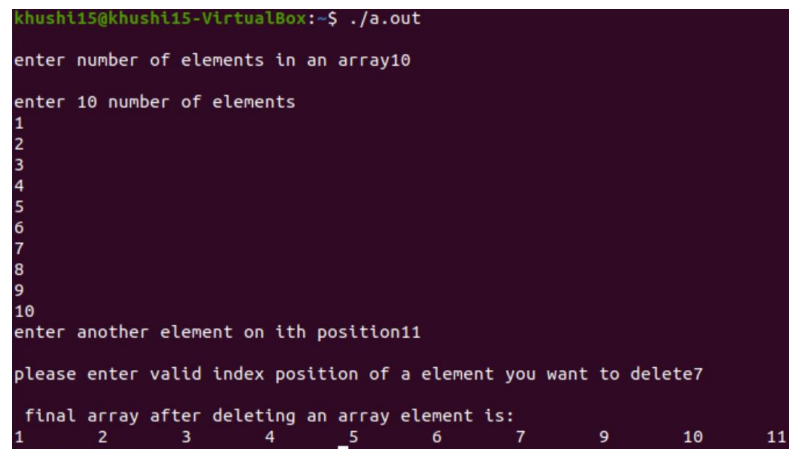
Step 8: Stop.**Code:**

```
#include<stdio.h>
int main()
{
    int a[20], position, i, size;
    printf("\nenter number of elements in an array");
    scanf("%d", &size);
    printf("\nenter %d number of elements\n",size);
    for(i=0;i<size;i++)
    {
        scanf("%d",&a[i]);
    }
    printf("enter another element on ith position");
    scanf("%d",&a[size]);
    printf("\nplease enter valid index position of a element
you want to delete");
    scanf("%d",&position);
    if(position<0 || position>=size)
        printf("\n enter a valid position between 0 and
%d",size);
    else
    {
        for(i=position;i<size;i++)
        {
            a[i]=a[i+1];
        }
        size--;
    }
    printf("\n final array after deleting an array element
is:\n");
```



```
for(i=0;i<=size;i++)
{
    printf("%d\t",a[i]);
}
return 0;
}
```

Output:



```
khushi15@khushi15-VirtualBox:~$ ./a.out
enter number of elements in an array10
enter 10 number of elements
1
2
3
4
5
6
7
8
9
10
enter another element on ith position11
please enter valid index position of a element you want to delete7
final array after deleting an array element is:
1    2    3    4    5    6    7    9    10   11
```

Output 32

Experiment 7.2

1. Convert uppercase string to lowercase using for loop.

Algorithm:

Step 1: Start

Step 2: Declare position, i, a[],size

Step 3: Read size.

Step 4: Run for loop to accept elements of array from user.

Step 5: Accept ith element from user

Step 6: Accept position of element from user and delete it from array using $a[i]=a[i+1]$ in for loop from $i=\text{position}$ to $i=\text{size}$.

Step 7: Display new array.

Step 8: Stop.

Code:

```
#include <stdio.h>
#define MAX_SIZE 100
int main()
{
    char str[MAX_SIZE];
    int i;
    printf("Enter any string: ");
    scanf("%s", str);
    for(i=0; str[i]!='\0'; i++)
    {
        if(str[i]>='A' && str[i]<='Z')
        {
            str[i] = str[i] + 32;
        }
    }
    printf("Lower case string: %s", str);
    return 0;
```

```
}
```

Test cases:

Input	Output
HELLO WORLD	hello world
Hello World	hello world
HeLlO wOrlD	hello world
Hello world	hello world

Output:

```
khushi15@khushi15-VirtualBox:~$ ./a.out
Enter a sentenceMy nAme Is KhushI
my name is khushikhushi15@khushi15-Virtu
```

Output 33

Experiment 8.1

1. Find the sum of rows and columns of matrix of given order.

Algorithm:

Step 1: Start

Step 2: Declare an array a and variables rows, columns, i, j and Sum

Step 3: Read elements of the array a

Step 4: FOR (rows=0, rows < i, rows++)

 FOR (columns=0, columns < j, columns ++)

 Sum=Sum +a[rows][columns]

 Print Sum

Step 5: FOR (columns=0, columns < i, columns++)

 FOR (rows=0, rows < j, rows ++)

 Sum=Sum +a[rows][columns]

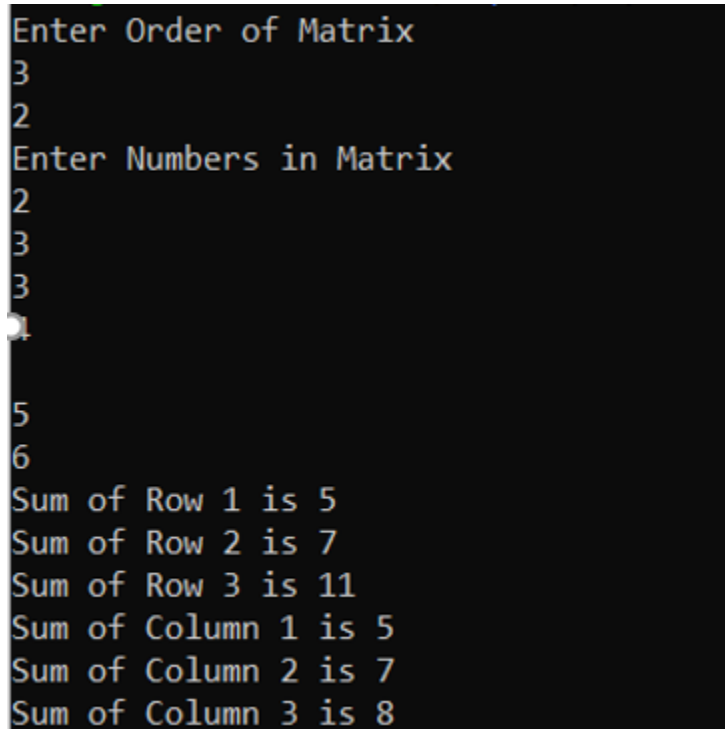
 Print Sum

Step 6: Stop

Code:

```
#include <stdio.h>
void main ()
{
    static int array[10][10];
    int i, j, m, n, sum = 0;
    printf("Enter Order of Matrix\n");
    scanf("%d %d", &m, &n);
    printf("Enter Numbers in Matrix\n");
    for (i = 0; i < m; ++i)
    {
        for (j = 0; j < n; ++j)
        {
            scanf("%d", &array[i][j]);
        }
    }
    for (i = 0; i < m; ++i)
    {
        for (j = 0; j < n; ++j)
        {
            sum = sum + array[i][j] ;
        }
    }
}
```

```
        printf("Sum of row %d is %d\n", i, sum);  
        sum = 0;  
    }  
    sum = 0;  
    for (j = 0; j < n; ++j)  
    {  
        for (i = 0; i < m; ++i)  
        {  
            sum = sum + array[i][j];  
        }  
        printf("Sum of column %d is %d\n", j, sum);  
        sum = 0;  
    }  
}
```

Output:

```
Enter Order of Matrix  
3  
2  
Enter Numbers in Matrix  
2  
3  
3  
4  
5  
6  
Sum of Row 1 is 5  
Sum of Row 2 is 7  
Sum of Row 3 is 11  
Sum of Column 1 is 5  
Sum of Column 2 is 7  
Sum of Column 3 is 8
```

Output 34

2. Count how many even numbers are there in a given integer array. [Hint: Linear Search]

Algorithm:

Step 1: Start

Step 2: Declare e_count, i and n

Step 3: Read n

Step 4: Declare an array[n]

Step 5: Read elements of the array[n]

Step 6: FOR (i=0; i < n; i++)

 IF (array[i]%2 == 0)

 e_count++

Step 7: Print e_count

Step 8: Stop

Code:

```
#include <stdio.h>
int main()
{
    int e_count=0,i,n;
    printf("Enter Number of Elements\n");
    scanf("%d",&n);
    int array[n];
    printf("Enter Elements in the array.\n");
    for(i=0;i<n;i++)
    {
        scanf("%d",&array[i]);
    }
    for(i=0;i<n;i++)
    {
        if(array[i]%2==0)
        {
            e_count++;
        }
    }
}
```

```
printf("Total Number of Even Numbers are %d\n",e_count);  
return 0;  
}
```

Test cases:

Input	Output
3 [0,1,2]	Total Number of Even Numbers are 2
4 [1,4,6,8]	Total Number of Even Numbers are 3
5 [1,3,5,7,9]	Total Number of Even Numbers are 0
3 [0,0,0]	Total Number of Even Numbers are 3

Output:

```
Enter Number of Elements  
5  
Enter Elements in the array.  
2  
3  
9  
7  
5  
Total Number of Even Numbers are 1
```

Output 35

Experiment 8.2

1. Store 'n' integers in an array in ascending or descending order. Search for a number with binary search technique.

Algorithm:

Step 1: Start

Step 2: Declare an array arr[] and variables i, j, temp and size

Step 3: Read size

Step 4: Read elements of the array arr[]

Step 5: Sort the array arr[] in ascending order

Step 6: Declare variables search, first, last and middle

Step 7: Read search

Step 8: first=0

last=size-1

middle= first+last/2

Step 9: WHILE (first <=last)

IF (arr[middle] < search)

first = middle + 1;

ELSE IF (arr[middle] == search)

Print search

ELSE

last = middle - 1;

middle = (first + last)/2;

IF (first > last)

Print "invalid search"

Step 10: Stop

Code:

```
#include<stdio.h>

int main()
{
    int c, first, last, middle, n, search, array[100];
    printf("Enter Size\n");
    scanf("%d",&n);
    printf("Enter numbers in Ascending Order\n");
    for (c = 0; c < n; c++)
        scanf("%d",&array[c]);
```



```
printf("Enter number to search\n");
scanf("%d", &search);
first = 0;
last = n - 1;
middle = (first+last)/2;
while (first <= last) {
    if (array[middle] < search)
        first = middle + 1;
    else if (array[middle] == search) {
        printf("Found\n");
        break;
    }
    else
        last = middle - 1;
    middle = (first + last)/2;
}
if (first > last)
    printf("Not found! %d is not present in the list.\n",
search);
return 0;
}
```

Test cases:

Input		Output
3	[0,1,2] 2	Found
5	[9,4,5,2,10] 7	Not found! 7 is not present in the list
4	[17,34,21,34] 34	Found

Output:

```
Enter Size
9
Enter numbers in Ascending Order
1
2
3
4
5
6
7
8
9
Enter number to search
5
Found
```

Output 36

Experiment 9

1. Design a structure 'product' to store the details of the product purchased like product name, price per unit, number of quantities purchased, and amount spent. Get the name, price per unit, and number of quantities of the product purchased. Calculate the amount spent on the product and then display all the details of the procured product using structure pointers.

Algorithm:

Step 1: Start

Step 2: Declare structure including char productname[30], int qty, float price, float amount.

Step 3: Read values using pointers.

Step 4: Print the values of pItem.

Step 5: Stop.

Code:

```
#include <stdio.h>
struct product
{
    char productname[30];
    int qty;
    float price;
    float amount;
};
int main()
{
    struct product itm;
    struct product *pItem;
    pItem = &itm;
    printf ("Enter Name of Product\n");
    gets (pItem->productname);
    printf ("Enter Price per Unit\n");
    scanf ("%f", &pItem ->price);
```

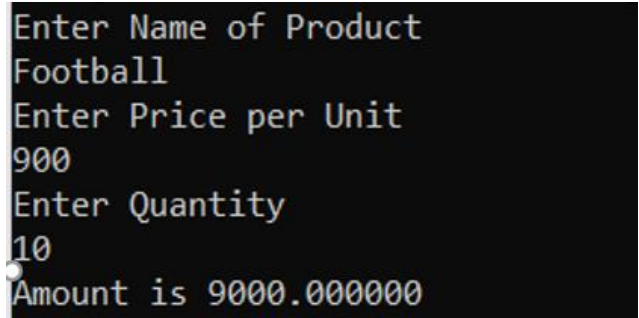
```

    printf ("Enter Quantity\n");
    scanf ("%d", &pItem->qty);
    pItem ->amount = (float) pItem->qty * pItem ->price;
    printf ("\nName: %s", pItem ->productname);
    printf ("\nPrice: %f", pItem ->price);
    printf ("\nQuantity: %d", pItem->qty);
    printf ("\nAmount is %f", pItem ->amount);
    return 0;
}

```

Test cases:

Input	Output
Pencil 2.50 10	Pencil 2.500000 10 25.000000
Eraser 3.00 20	Eraser 3.000000 20 60.000000
Sharpener 2.75 0	Sharpener 2.750000 0 0.000000

Output:


```

Enter Name of Product
Football
Enter Price per Unit
900
Enter Quantity
10
Amount is 9000.000000

```

Output 37

2. Design a structure 'student_record' to store student details like name, SAP ID, enrollment number, date of registration and date of birth. The element date of joining is defined using another structure 'date' to store date details like day, month, and year. Get data of 'n' students and then print the entered values [Hint: Use concept of Nested structures and Array of Structures].

Algorithm:

Step 1: Start.

Step 2: Declare structure date including int day, int month, int year.

Step 3: Declare structure student_record including char name[20], int SAP_ID, struct date birth, struct date registration.

Step 4: Accept name, SAP ID, date of birth and registration date from user.

Step 5: Display the above values after formatting.

Step 6: Stop.

Code:

```
#include <stdio.h>
int main()
{
    struct date
    {
        int day;
        int year;
        int month;
    };
    struct student_record
    {
        char name[20];
        int SAP_ID;
        struct date birth;
        struct date registration ;
    };
    struct student_record student[1000];
    int n, i;
    printf("Enter number of students:");
    scanf("%d", &n);
    for (i = 0; i<n; i++)
    {
        printf("Student name: \n");
        scanf("%s", &student[i].name);
        fflush(stdin);
        printf("SAP ID: \n");
        scanf("%d", &student[i].SAP_ID);
        fflush(stdin);
        printf("Date of Birth(dd-mm-yyyy): \n");
        scanf("%d-%d-%d",&student[i].birth.day,
&student[i].birth month, &student[i].birth.year);
        fflush(stdin);
        printf("Registration date(dd-mm-yyyy): \n");
```

```

scanf("%d-%d-%d", &student[i].registration.day,
&student[i].registration.month, &student[i].registration.year);
fflush(stdin);
}
printf("***** STUDENT RECORD
***** \n");
printf("-----\n");
printf("S.No. | NAME | SAP ID | Date of Birth |
Registration Date \n");
printf("-----\n");
for (i 0; i<n; i++)
{
printf("%d\t%s\t\t%d\t", i+1, student[i].name,
student[i].SAP_ID);
printf("%d/%d/%d\t", student[i].birth.day,
student[i].birth.month, student[i].birth.year);
printf("%d/%d/%d\n", student[i].registration.day,
student[i].registration.month, student[i].registration-year);
}
printf("-----\n");
}

```

Output:

```

Enter number of students:1
Student name:
Steve
SAP ID:
500091576
Date of Birth(dd-mm-yyyy):
20-09-2002
Registration date(dd-mm-yyyy):
01-07-2021

***** STUDENT RECORD *****
-----
S.No. | NAME | SAP ID | Date of Birth | Registration Date
-----
1 Steve 500091576 20/9/2002 1/7/2021
-----

```

Output 38

3. Design a union 'product' to store the details of the product purchased like product name, price per unit, number of quantities purchased, and amount spent. Get the name, price per unit, and

number of quantities of the product purchased. Calculate the amount spent on the product and then display all the details of the procured product using union pointers.

Algorithm:

Step 1: Start.

Step 2: Create a union named product.

Step 3: Inside the union, declare char name[20], int price, quantity & cost.

Step 4: Create union objects pro & *ptr.

Step 5: Create main function.

Step 6: Store address of pro in ptr.

Step 7: Read name, price & quantity.

Step 8: Calculate $\text{ptr} \rightarrow \text{cost} = \text{ptr} \rightarrow \text{quantity} * \text{ptr} \rightarrow \text{price}$.

Step 9: Display cost.

Step 10: Stop.

Code:

```
#include<stdio.h>
union product
{
    char name[50];
    int price;
    int quantity;
    int cost;
}pro,*ptr;
void main()
{
    ptr=&pro;
    printf("Enter name of product\n");
    scanf("%s",&pro.name);
    printf("Enter price\n");
    scanf("%s",&pro.price);
    printf("quantity\n");
    scanf("%s",&pro.name);
    printf("\nDetails\n");
    printf("Product name - %s\n",pro.name);
    printf("Price per Unit - %d\n",pro.price);
    printf("Total Quantity - %d\n",pro.quantity);
    ptr->cost=ptr->quantity*ptr->price;
    printf("Final cost - %d\n",pro.cost);
    return 0;
}
```

Output:

```
Enter Name of Product
Notebook
Enter Price
50
Enter Quantity
5
•
Details
Product Name -  📖
Price Per Unit - 5
Total Quantity - 5
Final Cost - 25
```

Output 39

Experiment 10

1. Design a structure 'subject' to store the details of the subject like subject name and subject code. Using structure pointer allocate memory for the structure dynamically so as to obtain details of 'n' subjects using for loop.

Algorithm:

Step 1: Start.

Step 2: Declare structure course including int subcode and char subject[30].

Step 3: Declare *ptr, noOfRecords.

Step 4: Accept number of records from user.

Step 5: Accept subject and subject code noOfRecord times through for loop from user.

Step 6: Display the records.

Step 7: Stop.

Code:

```
#include <stdio.h>
#include <stdlib.h>
struct course
{
    int subcode;
    char subject[30];
};

int main()
{
    struct course *ptr;
    int noOfRecords;
    printf("Enter Number of Subjects\n");
    scanf("%d", &noOfRecords);

    ptr = (struct course *)malloc(noOfRecords * sizeof(struct
course));
    for (int i = 0; i < noOfRecords; ++i)
    {
        printf("Enter Subject Name & Code of No.%d\n",i+1);
        scanf("%s %d", (ptr + i)->subject, &(ptr + i)->subcode);
    }

    printf("Details\n");
    for (int i = 0; i < noOfRecords; ++i) {
```

```
        printf("Subject Number %d\n", i+1);
        printf("- Name = %s\n- Code = %d\n", (ptr + i)->subject, (ptr
+ i)->subcode);
    }

    free(ptr);

    return 0;
}
```

Test cases:

Input

2 math 20 chem 50

Output

Details

Subject Number 1

- Name = math
- Code = 20

Subject Number 2

- Name = chem
- Code = 50

3 comp 03 phy 22 PE 34

Details

Subject Number 1

- Name = comp
- Code = 03

Subject Number 2

- Name = phy
- Code = 22

Subject Number 3

- Name = PE
- Code = 34

Output:

```
Enter Number of Subjects
2
Enter Subject Name & Code of No.1
Maths
1
Enter Subject Name & Code of No.2
PPL
2

Details

Subject Number 1
- Name = Maths
- Code = 1

Subject Number 2
- Name = PPL
- Code = 2
```

Output 40

2. Use self-referential structure to handle its elements with random and dynamically allocated memory.

Algorithm:

Step 1: Start
Step 2: Make structure of name node
Step 3: Declare variable data1, data2
Step 4: In main struct node ob1
Step 5: Initialization ob1.link = NULL; ob1.data1 = 10; ob1.data2 = 20;
Step 6: ob1.link = &ob2
Step 7: Accessing data members of ob2 using ob1
Step 8: Print the data
Step 9: Stop.

Code:

```
#include<stdio.h>
struct node
{
    int data1;
    char data2;
    struct node* link;
```

```
};  
  
int main()  
{  
  
    struct node ob1; // Node1  
    // Initialization  
    ob1.link = NULL;  
    ob1.data1 = 10;  
    ob1.data2 = 20;  
    struct node ob2; // Node2  
    // Initialization  
    ob2.link = NULL;  
    ob2.data1 = 30;  
    ob2.data2 = 40; // Linking ob1 and ob2  
    ob1.link = &ob2;  
    // Accessing data members of ob2 using ob1  
    printf("%d", ob1.link->data1);  
    printf("\n%d", ob1.link->data2);  
    return 0;  
}
```

Output: