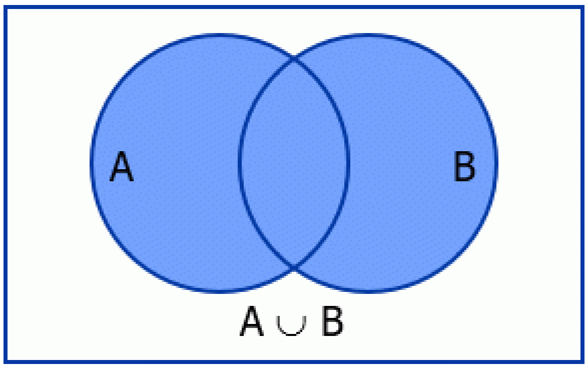
Lab 01
Title: To find the union of two numbers

Given two sets A and B, the union is the set that contains all elements or objects that belong to A and to B.

For example:

A = { **4, a**} and B = { b}

A U B = { 4, a, b }

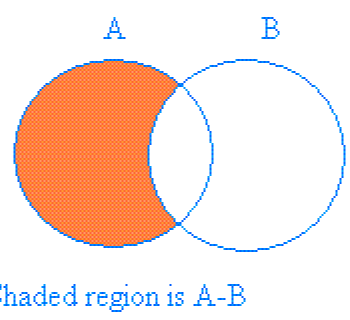


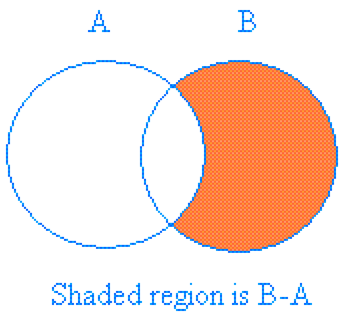
Program:
```
#include<stdio.h>
#include<conio.h>
int main()
{
   int a[10],b[10],i,c[10],j,k=0,n1,n2;
   // taking input set A
   printf("Enter number of  element of set A\n");
   scanf("%d",&n1);
   printf("Enter the element of set A \n");
   for(i=0;i<n1;i++)
      scanf("%d",&a[i]);
    // taking input set B
   printf("Enter number of element of set B\n");
   scanf("%d",&n2);
   printf("Enter the element of set B \n");
   for(i=0;i<n2;i++)
     scanf("%d",&b[i]);
   // logic for calculate union

   // copy the element of set A in set C
   for(i=0;i<n1;i++)
   {
       // repeted element is not allowed so we check is any value repeted
```

Lab 03

A. Title: To implement difference of sets

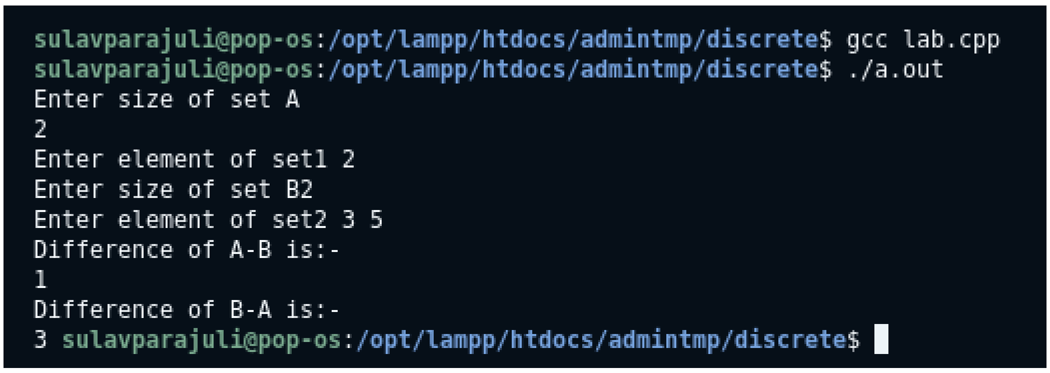
The difference between the sets A and B in this order is the set of elements which belong to A but not to B. Symbolically, we write A – B and read as " A minus B".





Program

```
#include<stdio.h>
int main()
{
    int a[10],b[10],c[10],d[10],m=0,k=0,n1,n2,l,i,j;
    printf("Enter size of set A");
    scanf("%d",&n1);
    printf("Enter element of set");
    for( i=0;i<n1;i++)
    scanf("%d",&a[i]);
    printf("Enter size of set B");
    scanf("%d",&n2);
    printf("Enter element of set");
    for( i=0;i<n2;i++)
    scanf("%d",&b[i]);
    // logic for find A-B
    for( i=0;i<n1;i++)
    {
        for(j=0;j<n2;j++)
        {
            if(b[j]==a[i])
             break;
        }
        if(j==n2)
        {
                for(l=0;l<k;l++)
            {
                if(c[l]==a[i])
                 break;
            }
            if(l==k)
            {
                c[k]=a[i];
                k++;
            }
        }

    }
    for( i=0;i<n2;i++)
    {
        for(j=0;j<n1;j++)
        {
            if(b[i]==a[j])
             break;
        }
        if(j==n1)
        {
          // here we check that is element already present in the set
          //if present than ignore it otherwise add to the difference set
            for(l=0;l<m;l++)
            {
                if(d[l]==b[i])
                 break;
            }
            if(l==m)
            {
                d[m]=b[i];
                m++;
            }
        }

    }
     printf("Difference of A-B is:-\n");
     for(i=0;i<k;i++)
     {
         printf("%d ",c[i]);
     }
     printf("\n");
     printf("Difference of B-A is:-\n");
     for(i=0;i<m;i++)
     {
         printf("%d ",d[i]);
     }
    return 0;
}
```



```
sulavparajuli@pop-os:/opt/lampp/htdocs/admintmp/discrete$ gcc lab.cpp
sulavparajuli@pop-os:/opt/lampp/htdocs/admintmp/discrete$ ./a.out
Enter size of set A
2
Enter element of set1 2
Enter size of set B2
Enter element of set2 3 5
Difference of A-B is:-
1
Difference of B-A is:-
3 sulavparajuli@pop-os:/opt/lampp/htdocs/admintmp/discrete$
```

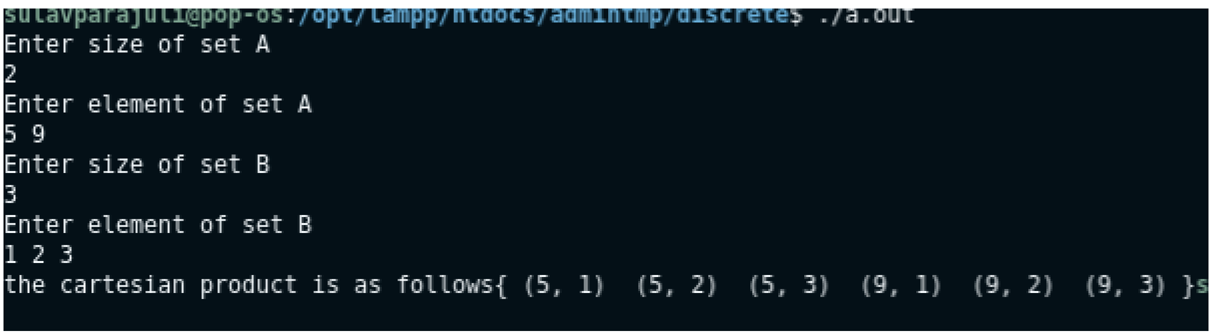
Lab 03

B. To find cartesian product of two set

A cartesian product of two non-empty sets A and B is the set of all possible ordered pairs where the first component of the pair is from A, and the second component of the pair is from B. A × B = {(a, b) : a ∈ A and b ∈ B}

Program

```
#include<stdio.h>
int main()
{
    int a[10],b[10],n1,n2;
    printf("Enter size of set A\n");
    scanf("%d",&n1);
    printf("Enter element of set A\n");
    for(int i=0;i<n1;i++)
      scanf("%d",&a[i]);
    printf("Enter size of set B\n");
    scanf("%d",&n2);
    printf("Enter element of set B\n");
    for(int i=0;i<n2;i++)
      scanf("%d",&b[i]);
    printf("the cartesian product is as follows");
    printf("{");
    for(int i=0;i<n1;i++)
    {
        for(int j=0;j<n2;j++)
        {
            printf(" (%d, %d) ",a[i],b[j]);
        }
    }
    printf("}");
    return 0;
}
```



```
sulavparajuli@pop-os:/opt/lampp/htdocs/admintmp/discrete$ ./a.out
Enter size of set A
2
Enter element of set A
5 9
Enter size of set B
3
Enter element of set B
1 2 3
the cartesian product is as follows{ (5, 1)  (5, 2)  (5, 3)  (9, 1)  (9, 2)  (9, 3) }s
```

Lab 04

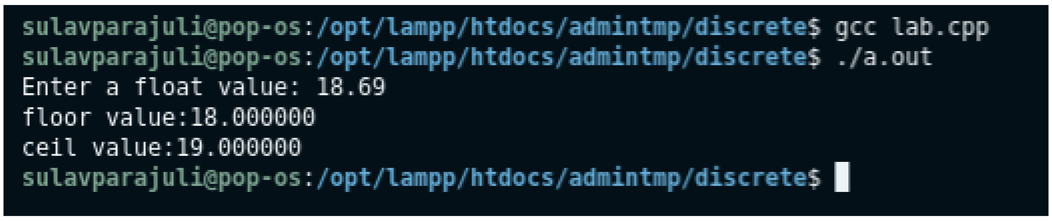
A. Program to implement floor and ceiling function

In mathematics and computer science, the **floor function** is the function that takes as input a real number x, and gives as output the greatest integer less than or equal to x, denoted $\operatorname{floor}(x)$ or $\lfloor x \rfloor$. Similarly, the **ceiling function** maps x to the least integer greater than or equal to x, denoted $\operatorname{ceil}(x)$ or $\lceil x \rceil$

For example, $\lfloor 2.4 \rfloor = 2$, $\lfloor -2.4 \rfloor = -3$, $\lceil 2.4 \rceil = 3$, and $\lceil -2.4 \rceil = -2$.

Progarm

```
#include <stdio.h>
#include <math.h>
 int main()
{
    float val;
    float fVal,cVal;
    printf("Enter a float value: ");
    scanf("%f",&val);

    fVal=floor(val);
    cVal =ceil(val);
    printf("floor value:%f \nceil value:%f\n",fVal,cVal);
    return 0;
}
```



```
sulavparajuli@pop-os:/opt/lampp/htdocs/admintmp/discrete$ gcc lab.cpp
sulavparajuli@pop-os:/opt/lampp/htdocs/admintmp/discrete$ ./a.out
Enter a float value: 18.69
floor value:18.000000
ceil value:19.000000
sulavparajuli@pop-os:/opt/lampp/htdocs/admintmp/discrete$ 
```

Lab 04
B. **Program to find GCD of two numbers using Euclidean Algorithm**

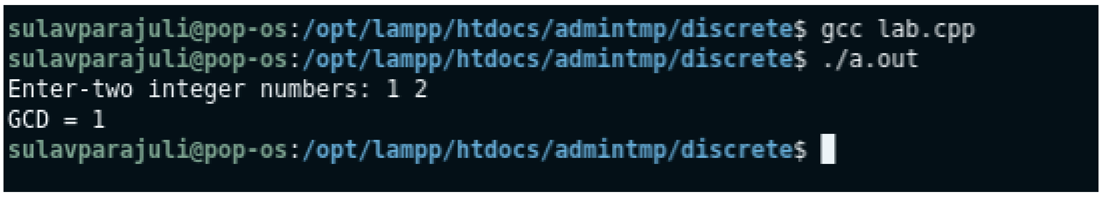
Basic Euclidean Algorithm for GCD

The algorithm is based on the below facts.

- If we subtract a smaller number from a larger (we reduce a larger number), GCD doesn't change. So if we keep subtracting repeatedly the larger of two, we end up with GCD.
- Now instead of subtraction, if we divide the smaller number, the algorithm stops when we find remainder 0.

Program

```
 #include <stdio.h>
int main()
{
   int m, n; /* given numbers */
   printf("Enter-two integer numbers: ");
   scanf ("%d %d", &m, &n);
   while (n > 0) {
      int r = m % n;
      m = n;
      n = r;
   }
   printf ("GCD = %d \n",m);
   return 0;
}
```



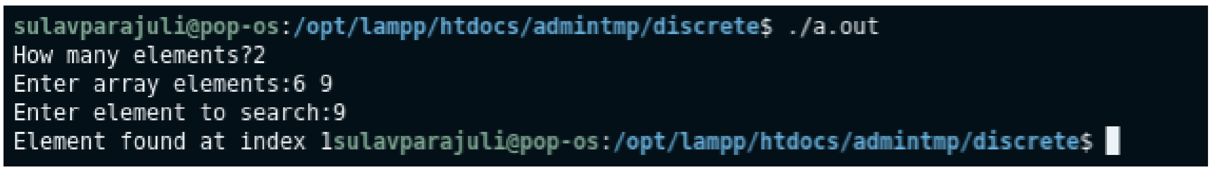
```
sulavparajuli@pop-os:/opt/lampp/htdocs/admintmp/discrete$ gcc lab.cpp
sulavparajuli@pop-os:/opt/lampp/htdocs/admintmp/discrete$ ./a.out
Enter-two integer numbers: 1 2
GCD = 1
sulavparajuli@pop-os:/opt/lampp/htdocs/admintmp/discrete$
```

Lab 05
Title: To Implement Linear Search

A linear search or sequential search is **a method for finding an element within a list**. It sequentially checks each element of the list until a match is found or the whole list has been searched.

Program

```
#include<stdio.h>
 int main()
{
   int a[20],i,x,n;

   printf("How many elements?");
   scanf("%d",&n);

   printf("Enter array elements:");
   for(i=0;i<n;++i)
      scanf("%d",&a[i]);

   printf("Enter element to search:");
   scanf("%d",&x);

   for(i=0;i<n;++i)
      if(a[i]==x)
         break;

   if(i<n)
      printf("Element found at index %d",i);
   else
      printf("Element not found");

   return 0;
}
```



```
sulavparajuli@pop-os:/opt/lampp/htdocs/admintmp/discrete$ ./a.out
How many elements?2
Enter array elements:6 9
Enter element to search:9
Element found at index 1sulavparajuli@pop-os:/opt/lampp/htdocs/admintmp/discrete$
```

Lab 06
Title: To perform Merge Sort using Recursion

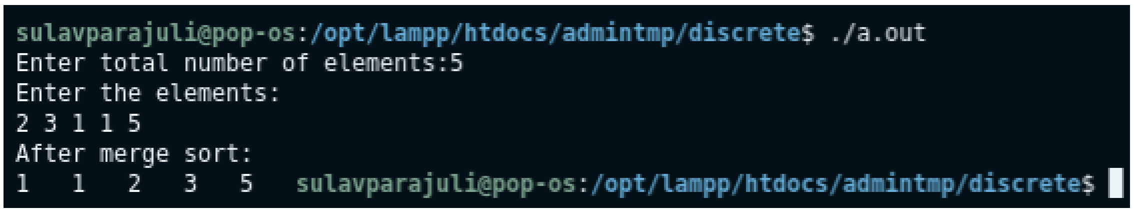
Merge Sort is one of the most popular sorting algorithms that is based on the principle of Divide and Conquer Algorithm.

Here, a problem is divided into multiple sub-problems. Each sub-problem is solved individually. Finally, sub-problems are combined to form the final solution.

Program

```
#include <stdio.h>

void mergeSort(int [], int, int, int);
void partition(int [],int, int);

int main()
{
    int list[50];
    int i, size;

    printf("Enter total number of elements:");
    scanf("%d", &size);
    printf("Enter the elements:\n");
    for(i = 0; i < size; i++)
    {
        scanf("%d", &list[i]);
    }
    partition(list, 0, size - 1);
    printf("After merge sort:\n");
    for(i = 0;i < size; i++)
    {
        printf("%d   ",list[i]);
    }

   return 0;
}

void partition(int list[],int low,int high)
```

```
{
    int mid;

    if(low < high)
    {
        mid = (low + high) / 2;
        partition(list, low, mid);
        partition(list, mid + 1, high);
        mergeSort(list, low, mid, high);
    }
}

void mergeSort(int list[],int low,int mid,int high)
{
    int i, mi, k, lo, temp[50];

    lo = low;
    i = low;
    mi = mid + 1;
    while ((lo <= mid) && (mi <= high))
    {
        if (list[lo] <= list[mi])
        {
            temp[i] = list[lo];
            lo++;
        }
        else
        {
            temp[i] = list[mi];
            mi++;
        }
        i++;
    }
    if (lo > mid)
    {
        for (k = mi; k <= high; k++)
        {
            temp[i] = list[k];
            i++;
        }
    }
    else
    {
        for (k = lo; k <= mid; k++)
        {
            temp[i] = list[k];
            i++;
        }
    }

    for (k = low; k <= high; k++)
    {
        list[k] = temp[k];
    }
}
```



```
sulavparajuli@pop-os:/opt/lampp/htdocs/admintmp/discrete$ ./a.out
Enter total number of elements:5
Enter the elements:
2 3 1 1 5
After merge sort:
1   1   2   3   5   sulavparajuli@pop-os:/opt/lampp/htdocs/admintmp/discrete$
```