# Experiment No 1

**AIM: Implementation of Set Theory Problem (Cricket, Badminton and Football) OBJECTIVES:**

* Tounderstandtheconceptoflistsinpython.
* Toexploretheconceptofsettheory.

PROBLEM STATEMENT:

Insecondyearcomputerengineeringclass,groupAstudent’splaycricket,groupB studentsplaybadmintonandgroupCstudentsplayfootball.

Write a Python program using functions to compute following: -

1. Listofstudentswhoplaybothcricketandbadminton
2. Listofstudentswhoplayeithercricketorbadmintonbutnotboth
3. Numberofstudentswhoplayneithercricketnorbadminton
4. Numberofstudentswhoplaycricketandfootballbutnotbadminton.

(Note-Whilerealizingthegroup,duplicateentriesshouldbeavoided,DonotuseSET built-infunctions

OUTCOMES:

* Usealgorithmsonvariouslineardatastructureusingsequentialorganizationtosolve real lifeproblems.

SOFTWARE & HARDWARE REQUIREMENTS:

Softwares: Open Source Python, Programming tool like Google Colab, Jupyter Notebook, Pycharm, Spyder, G++/GCC

Hardware: Pentium Dual Core (3.00GHz)

4 GB RAM

THEORY:

**Settheory**isthebranchofmathematicallogicthatstudiessets,whichinformallyare collectionsofobjects.Althoughanytypeofobjectcanbecollectedintoaset,settheoryis appliedmostoftentoobjectsthatarerelevanttomathematics.Thelanguageofsettheorycan beusedinthedefinitionsofnearlyallmathematicalobjects.

Definitions:

* 1. **Equalsets:**WedefineA=BifAandBhavethesameelements.
  2. **Subset:**WesaythatAisasubsetofBandwewriteA⊂BorB⊃Aifevery elementofAisalsoanelementofB.(WealsosaythatAisincludedinBorBincludesAor BisasupersetofA.)
  3. **Propersubset:**WesaythatAisapropersubsetofBandwewriteA⊂Bstrictlyif A⊂BandA≠B.(Thereexistsatleastoneelementb∈Bsuchthatb∉A.)
  4. **Theemptyset:**Thesetwhichhasnoelementiscalledtheemptysetandisdenoted by∅.(Thatis∅={x∈A:x∉A},whereAisanyset.)
  5. **Powersetofaset:**LetXbeanyset.ThesetofallsubsetsofXiscalledthepower setofXandisdenotedbyP(X).(ThatiswedefineP(X):={A:A⊂X})

Remarks:

Let A and B be any sets. Then the following propositions can be proved easily:

1. A=BifandonlyifA⊂BandB⊂A,
2. A⊂Aand∅⊂A,
3. A∈P(A)and∅∈P(A),
4. P(∅)={∅}.(P(∅)isnotempty,ithasexactlyoneelement,the∅.)

OPERATIONS BETWEEN SETS

LetHbeasetincludingallsetsA,B,C,…whichoccurinthefollowing.LetuscallH the basicset.

**Union of sets:** (denoted by ∪, called ”union” or ”cup”)

1. TheunionofsetsAandBisdefinedbyA∪B:={x∈H:x∈Aorx∈B}.
2. TheunionofasetAofsetsisdefinedbySA:={x∈H:∃A∈Ax∈A}.(xbelongs toatleastoneelementofA)

**Intersection of sets:** (denoted by ∩, called”intersection” or”cap”)

1. TheintersectionofsetsAandBisdefinedbyA∩B:={x∈H:x∈Aandx∈B}.
2. TheintersectionofasetA6=∅isdefinedbyTA:={x∈H:∀A∈Ax∈A}.(x belongstoallelementsofA)

**Definition** (Disjoint sets.):

A and B are called disjoint sets if A ∩ B = ∅ (they have no elements in common).

**Difference of sets**: (denoted by \ )

1. ThedifferenceofsetsAandBisdefinedbyA\B:={x∈H:x∈Aandx/∈B}.(We alsosaythatA\BisthecomplementofBwithrespecttoA.)
2. H\BiscalledthecomplementofBandisdenotedbyBc,thatisBc:={x∈H:x/∈

B}.

**Orderedpairs:**Letxandybeanyobjects(e.g.anyelementsofthebasicsetH).The orderedpair(x,y)isdefinedby(x,y):={{x},{x,y}}.Wecallxandythefirstandthe secondcomponentsoftheorderedpair(x,y),respectively.Incasex=ywehave(x,x)={{x}

} .

**Cartesianproductofsets:**LetAandBbesets.TheCartesianproductofAandBis definedbyA×B:={(a,b):a∈Aandb∈B},i.e.theCartesianproductA×Bisthesetof allorderedpairs(a,b)witha∈A,b∈B.

Examples:

1. A ∪ B = { 1, 2, 3}



2. A ∩ B = { 2 }



3. Ac or ~A= {3, 4}



4. A – B= {1 }



5. ~(A U B )= {4}



6. ~(A ^ B)or~( }= {1, 3,4}



**CONCLUSION:** Thus we have implemented python program for set operations

# Experiment No 2

**Title:** Write a Python program to compute different operations on String.

**Objectives:**

**•** To understand the use standard library functions for string operations

• To accepts string/statements from user

• To perform the string operations.

**Problem Statement: -** Write a Python program to compute following operations on String:

a) To display word with the longest length

b) To determines the frequency of occurrence of particular character in the string

c) To check whether given string is palindrome or not

d) To display index of first appearance of the substring

e) To count the occurrences of each word in a given string.

**Outcome:**

**•** Display string/statements

• Find and display longest length of word, palindrome of string, occurrences of character, find 1stindex position of substring.

### How to create a string in Python

* Strings can be created by enclosing characters inside **single quotes** or **double-quotes**.
* Triple quotes can also be used in Python, but are generally used to represent multi-line strings and docstrings.
* *# string with single quotes*
* my\_string = 'Welcome'
* print(my\_string)
* *# string with double quotes*
* my\_string = "Welcome I’m in Strings"
* print(my\_string)
* *# string with triple quotes*
* my\_string = '''Welcome'''
* print(my\_string)

### Indexing in Strings

* We can access individual characters using indexing or a range of characters using slicing.
* Index will always start from 0.
* Trying to access a character out of index range will raise an IndexError.
* The index must be an integer.

### Accessing Values in String

To access each value or sub-string, use the square brackets to slice along the index or indices to obtain your sub-string.

*#Accessing string characters in Python*

str1 = 'Computer'

print('str1 = ', str1)

*#string are immutable*

*# str1[0] ='c'*

*#first character*

print('str1[0] = ', str1[0])

*#last character*

print('str1[-1] = ', str1[-1])

*#index Error*

*#print('str1[-1] =', str1[9])*

*#slicing 2nd to 5th character*

print('str1[3:5] = ', str1[3:5])

*#slicing can be done by slice function*

x=slice(3,5)

print('str1[3,5]= ', str1[x])

*#slicing 6th to 2nd last character*

print('str1[5:-2] = ', str1[5:-2])

* Joining two or more strings into a single string is called concatenation.
* The + operator will be used to concatenate in Python.
* The \* operator can be used to repeat the string for a given number of times.

A string object is one of the sequence data types in Python. It is an immutable

sequence of Unicode characters. Strings are objects of Python's built-in class 'str'.

String literals are written by enclosing a sequence of characters in single quotes

('hello'), double quotes ("hello") or triple quotes ('''hello''' or """hello""").

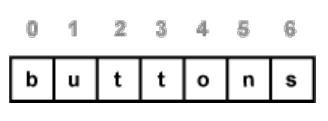
Example :- str1==I am the student of AVCOE=

str2==Hello, everyone=

Various Types of operation performing on strings :-

* 1. Indexing To string :- The beginning character of a string corresponds to index 0 and the last character corresponds to the index

Index start with <0=



In the string there are two ways to represent the index of string. Firstly index start

with <0= and increasing right to left one by one . Another one is from left to right

the last element have index by default <-1=.

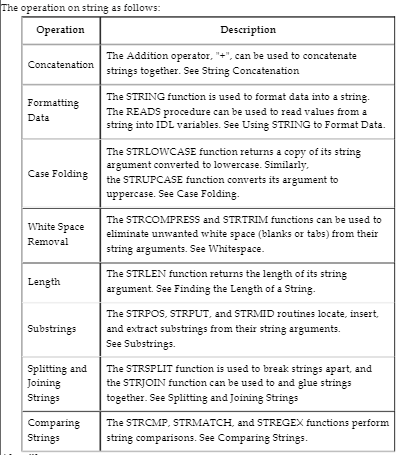
Palindrome string:-

**Palindrome string**:A palindrome is a string that is the same read forward or backward. For

example, "dad" is the same in forward or reverse direction. Another example is

"aibohphobia", which literally means, an irritable fear of palindromes. Simply the

word read from right to left or left to right is also same.



**Algorithm:**

1) Display word with the longest length :-

#To display word with the longest length

str1=input("Enter any string:")

list1=str1.split()

m=0

word=0

print(list1)

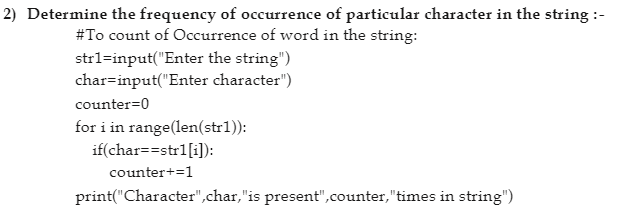
for i in range(len(list1)):

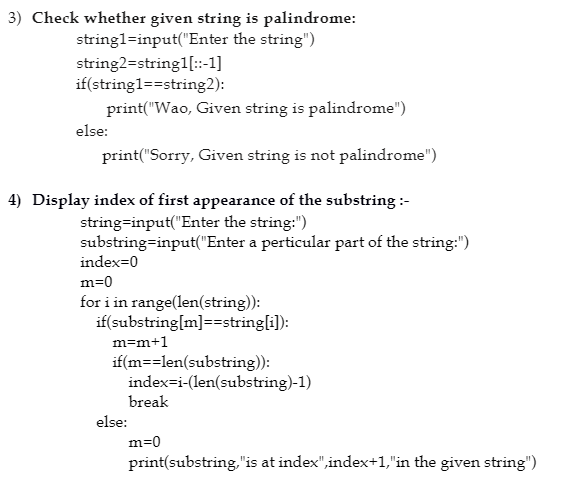
if m<len(list1[i]):

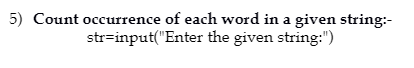
m=len(list1[i])

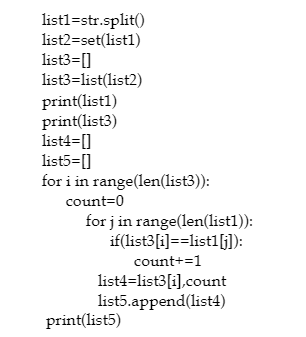
word=i

print("Word with longest length is:",len(list1[word]))









# Experiment No. 3

**Title**:MatrixOperations.

## Objectives:

* Tounderstandimplementationoftwodimensionalarray.
* UnderstandtheimplementationofMatrixandperformvariousoperationsonitusingtwodimensionalarrays.

## ProblemStatement:

WriteC/C++programforstoringmatrix.Writefunctionsfor

1. Checkwhethergivenmatrixis uppertriangularornot
2. Computesummationofdiagonalelements
3. Computetransposeofmatrix
4. Add,subtractandmultiplytwomatrices
5. Determines the location of a saddle point if one exists (An m x n matrix is said to have a saddle point ifsomeentrya[i][j]is thesmallestvalue inrowi andthelargestvalue inj.)

## Outcomes:

Oncompletionofthisassignmentstudentswill beable to-

* + Implementthetwo dimensionalarrays.
  + Solverealworldproblemofmatrixandperformvariousoperationsonitlogicallyusingtwodimensionalarrays.

## Software&Hardwarerequirements:

* + OpenSourceCProgrammingtoolslikeG++/GCCorEclipse.
  + 64-bitOpensourceLinuxoritsderivative.

## Theory-Concept:

**2DimensionalArrays:**

* Two-dimensionalarrayarethosetypeofarray,whichhasfinitenumberofrowsandfinitenumberof columns.Thedeclaration formof2-dimensionalarray is

# Data\_type Array\_name[rowsize][columnsize];

* Thetypemaybe anyvalidtype supported byC.
* Theruleforgivingthearray nameissameastheordinaryvariable.
* Therowsizeandcolumnsizeshouldbeanindividualconstant.

The following declares a two-dimensional 3 by 3 array of integers and sets the first and last elementstobe10.

# int matrix[3][3];

**matrix[0][0]=10;**

# matrix[2][2]=10;

The following Figure illustrates a two dimensional array, matrix. The array contains three rowsand tree columns, so it is said to be a 3-by-3 array. In general, an array with m rows and n columns iscalledanm-by-narray.

|  |  |  |  |
| --- | --- | --- | --- |
|  | [0] | [1] | [2] |
| [0] | 10 |  |  |
| [1] |  |  |  |
| [2] |  |  | 10 |

Every element in array matrix is identified by an element name of the form matrix[ i ][ j ]; matrixis the name of the array, and i and j are the subscripts that uniquely identify each element in matrix .Notice that the names of the elements in the first row all have a first subscript of 0; the names of theelementsinthethird column all haveasecond subscript of2.

In the case of Two-dimensional array, during declaration the maximum number of rows andmaximumnumberofcolumn shouldbe specifiedforprocessingallarrayelements.

The implementation of the array stores all the elements in a single contiguous block of memory.The other possible implementation would be a combination of several distinct one-dimensional arrays.That’s not how C does it. In memory, the array is arranged with the elements of the rightmost index nexttoeachother.Inother words,matrix[1][1] comes rightbeforematrix[1][2]inmemory.

Thefollowingarray:

|  |  |  |  |
| --- | --- | --- | --- |
|  | [0] | [1] | [2] |
| [0] | 1 | 2 | 3 |
| [1] | 4 | 5 | 6 |
| [2] | 7 | 8 | 9 |

## wouldbestored:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

**SAMPLE PROGRAM for Two Dimensional Array to store and display values:#include<conio.h>**

## #include<stdio.h>intmain()

**{**

## int matrix [3][3],i,j,r,c;clrscr();

**printf(“Enter the order of matrix\n”);scanf(“%d%d”,&r,&c);**

## printf(“Enter elements of %d \* %d matrix \n”,r,c);for(i=0;i<r;i++)

**for(j=0;j<c;j++)**

## scanf(“%d”,&matrix[i][j]);printf(“Givenmatrix:\n”);

**for(i=0;i<r;i++)**

## {

**for(j=0;j<c;j++)**

## printf(“%d\t”,matrix[i][j]);

**printf(“\n”);**

## }

**}**

## Output;

**printf(“%d\t”,matrix[2][2]);getch();**

## return0;

**Enter the order of matrix2**

## 2

**Enter elements of 2\*2 matrix1**

## 2

**3**

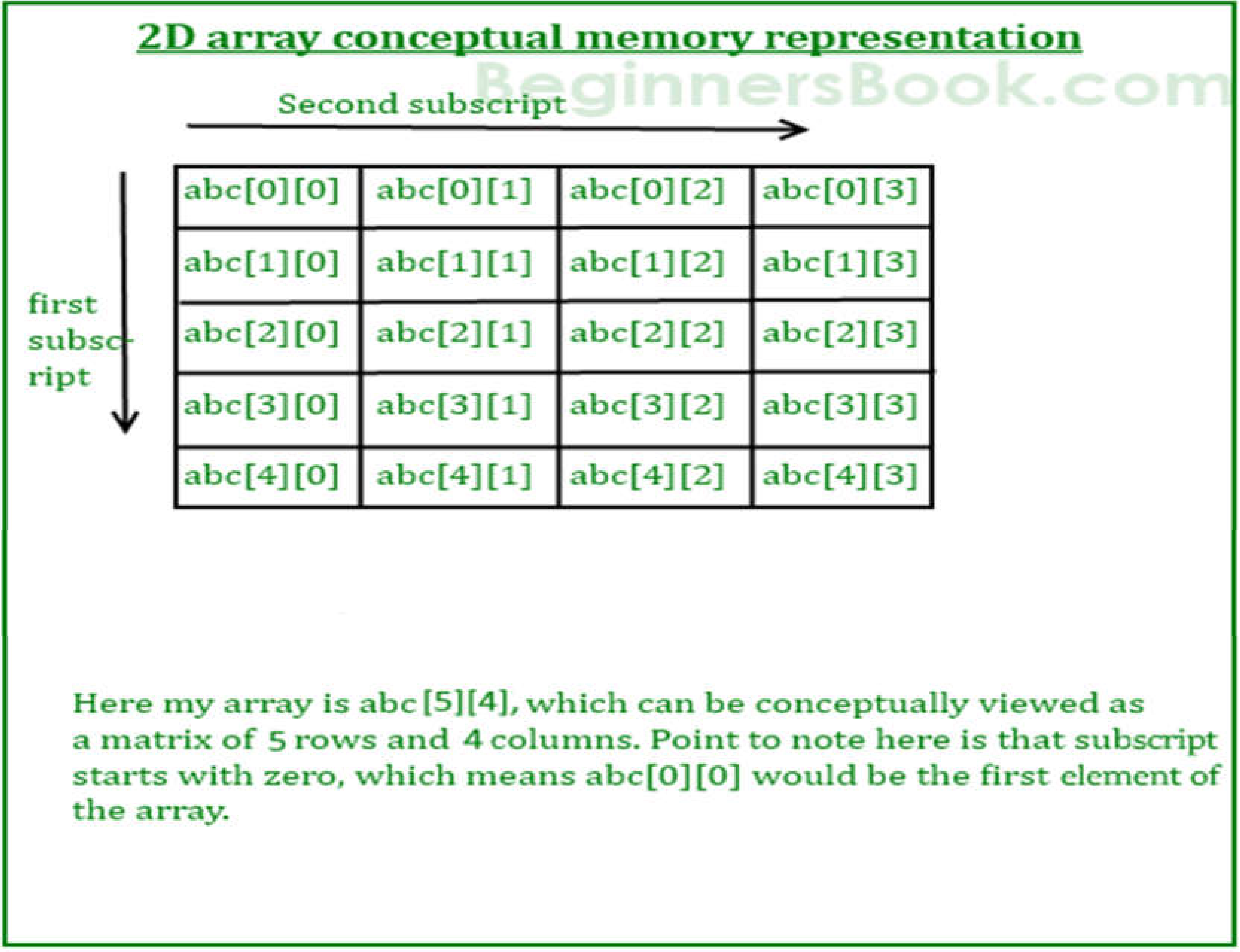
## 4

**Givenmatrix:**

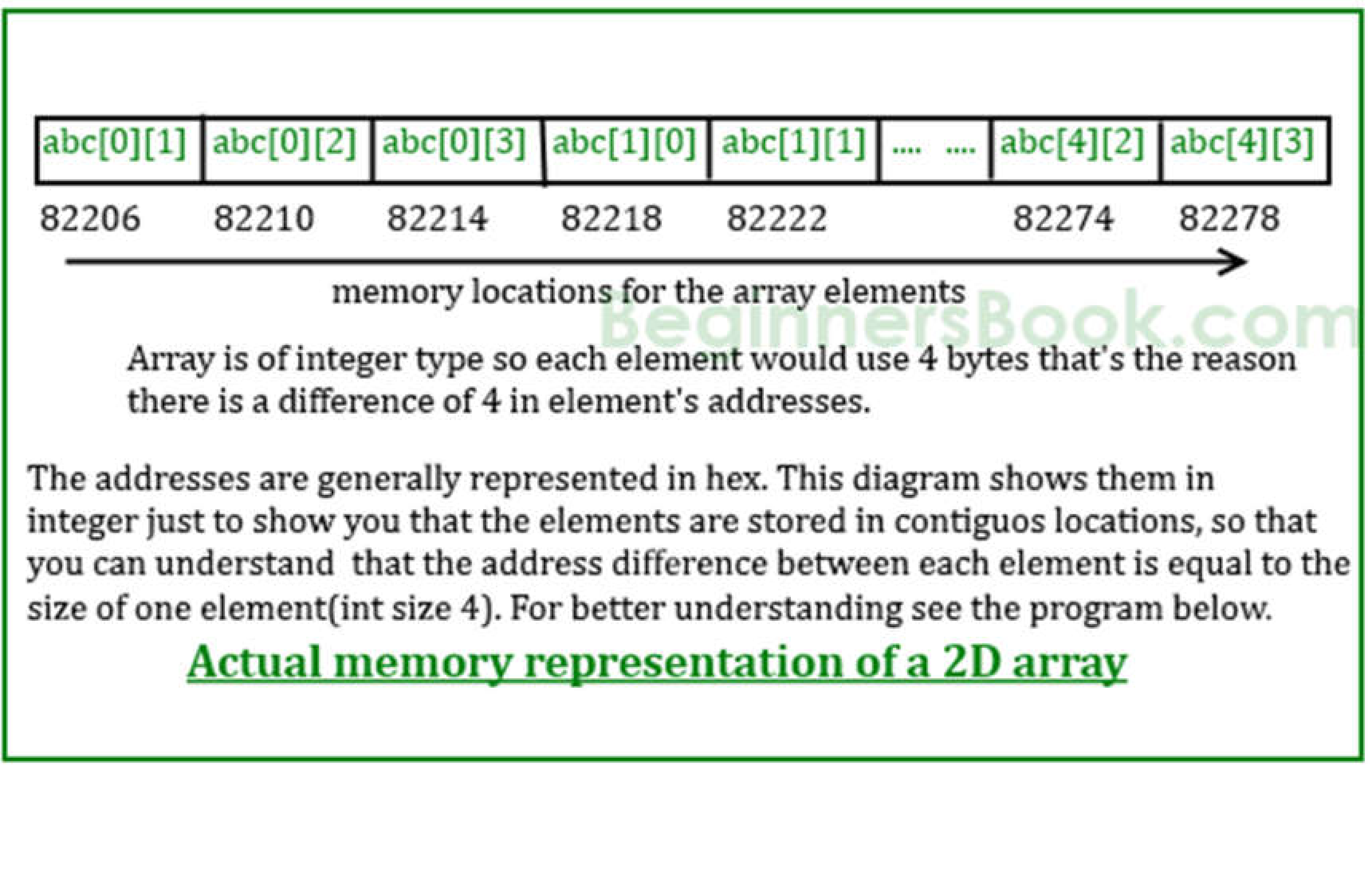
## 1 2

**3 4**

Inbelowexample,Ihavea2Darrayabcofintegertype.Conceptuallyyoucanvisualizetheabovearraylikethis:



Howevertheactualrepresentation ofthisarrayinmemorywouldbesomethinglikethis:



## Algorithm:

1. **Toperformthe additionoftwomatrices**

## Description:Theprogramtakesthetwomatrixesof samesize andperforms theadditionAlgorithm:

Step1:start

Step 2: read the size of matrices A,B – m,nStep3:readtheelementsofmatrixAStep4:readtheelementsofmatrixBStep 5: perform the addition operationStep6: print sumofmatrices Aand B

Step7:Stop

## Calculatingtransposeof amatrixin-placemanner.

**DESCRIPTION:The transposeofa matrixisobtainedbyinterchangingthe rhence the orderoftheresultantmatrix changes.**

## ALGORITHM:

Step1:start

Step2: readthesizeofmatrixA

Step3:readtheelementsofmatrixA

Step4:performthetransposeoperationbyinterchangingtherowand columnvalues,theorderof theresultantmatrix

number of rows in transpose matrix=number of columns in the given matrixnumber of columns in transpose matrix=number of rows in the given matrixStep5: Transposeisobtained through

at[i][j]=a[i][j]

Step 6: print the resultant transpose matrix at.Step7: stop

## Matrixmultiplicationbycheckingcompatibility

**DESCRIPTION:Takesthetwomatrixesofdifferentsizesandchecksforpossibilityofmultiplicationandperform multiplicationifpossible. ALGORITHM:**

Step1:start

Step2:readthesizeofmatricesA,B

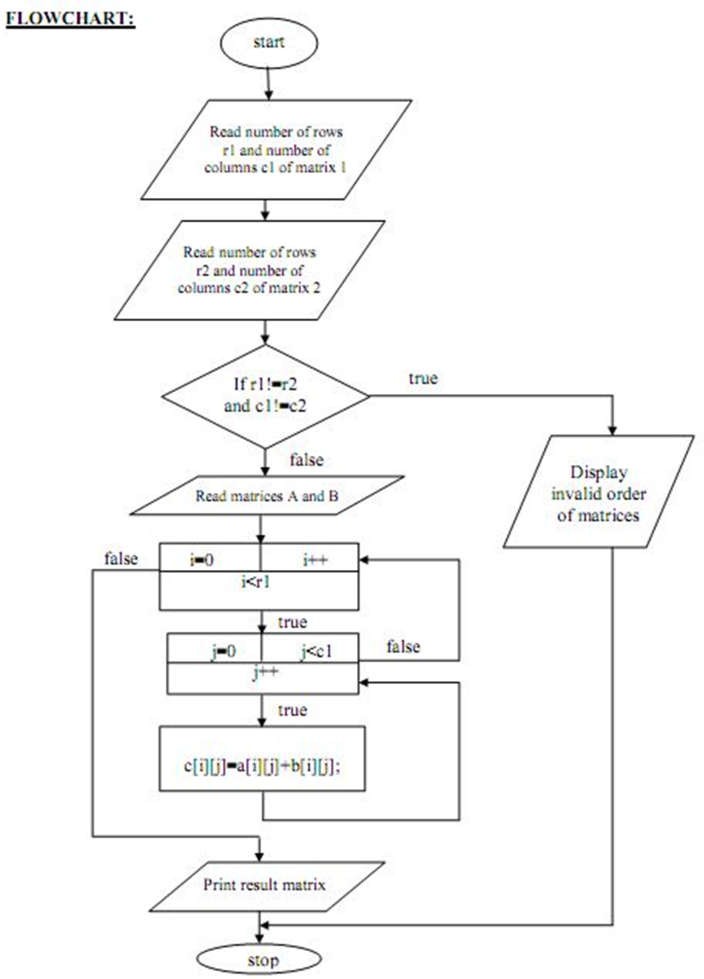
Step3:checkcompatibilityofmatricesformultiplicationi.e,numberofcolumns inthefirstmatrixshould beequal tonumberofrowsin thesecond matrix.

Step 4: read the elements of matrix AStep5:readtheelementsofmatrixB

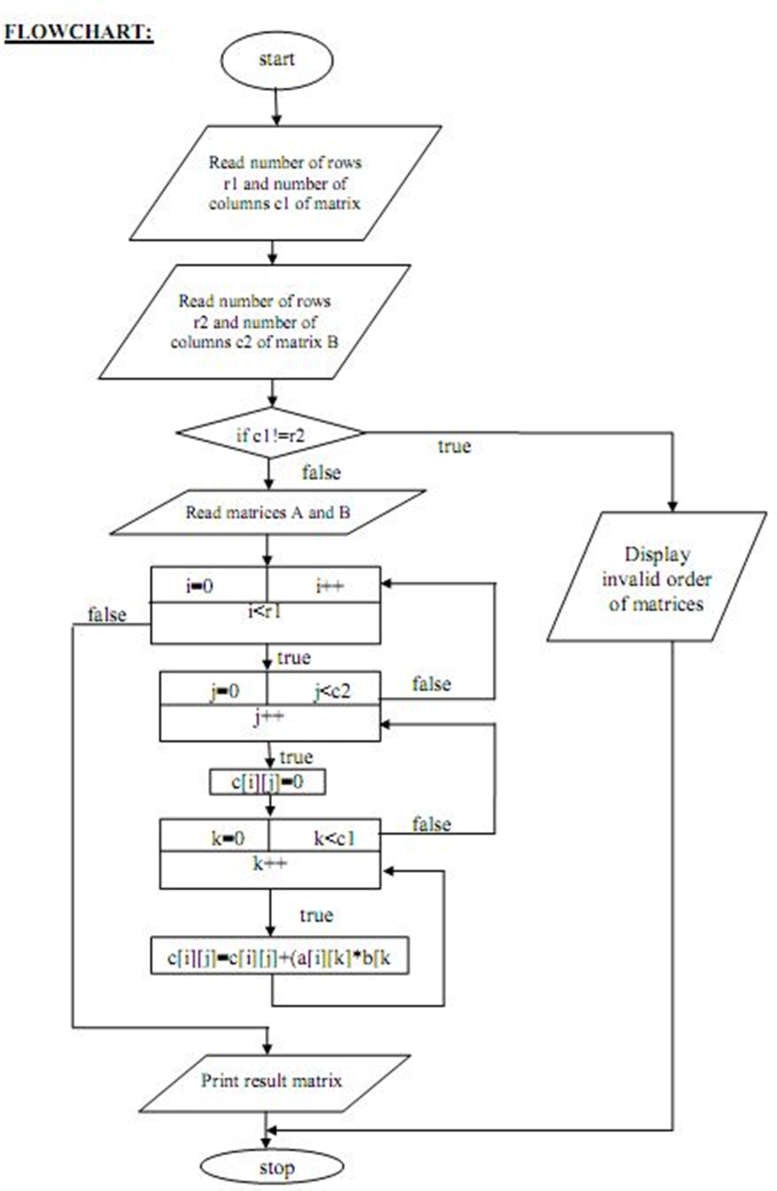
Step 6: perform the multiplication operation by storing the resulting values into matrix C.Step7: print the resultantmatrixC.

Step8:Stop

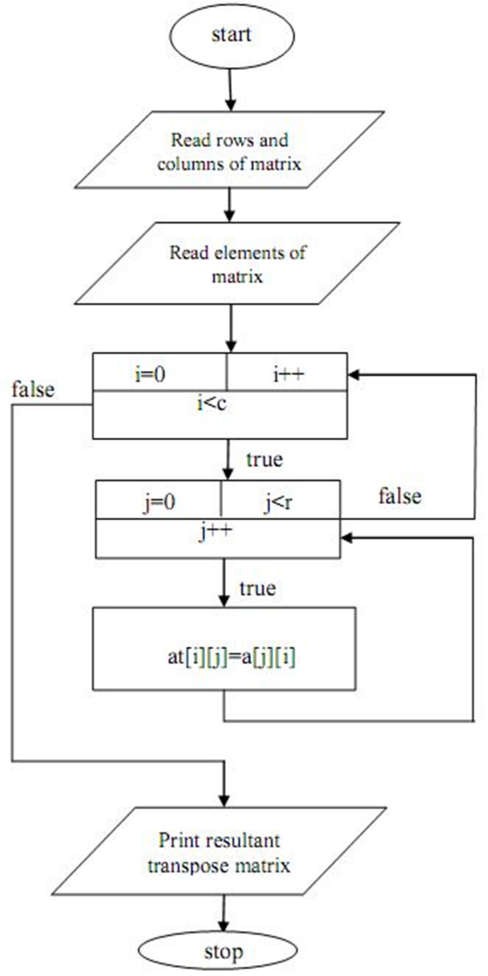
## Flowchart:



**Fig.1Flowchart forAdditionofTwoMatrices**



## Fig.2FlowchartforMultiplicationofTwoMatrices



**Fig.3FlowchartforTransposeofMatrix**

## Conclusion:

HencewehavelearnedTwoDimensionalArrayasaDatastructureandhowtouseittoimplementmatrix.

# Experiment No 4

AIM:Implementationofsortingmethods1.Selectionsort2.Bubblesort OBJECTIVES:

* Tounderstandtheconceptofsortingmechanism.
* Toexplorethedeepconceptoftimecomplexitiesofdifferentsortingtechniques.

PROBLEM STATEMENT:

Writepythonprogramtostorefirstyearpercentageofstudentsinarray.Writefunctionfor sortingarrayoffloatingpointnumbersinascendingorderusing

1. SelectionSort
2. Bubblesortanddisplaytopfivescores.

OUTCOMES:

 Toanalyze&understandvarioussearching&sortingalgorithm.

SOFTWARE & HARDWARE REQUIREMENTS:

Softwares: Open Source Python, Programming tool like Google Colab, Jupyter Notebook, Pycharm, Spyder, G++/GCC

Hardware: Pentium Dual Core (3.00GHz) 4 GB RAM

THEORY:

**Selection Sort :**

Selectionsortisasimplesortingalgorithm.Thissortingalgorithmisain-placecomparison basedalgorithminwhichthelistisdividedinto twoparts,sortedpartatleftendand unsortedpartatrightend.Initiallysortedpartisemptyandunsortedpartisentirelist.

Smallestelementisselectedfromtheunsortedarrayandswappedwiththeleftmostelement andthatelementbecomespartofsortedarray.Thisprocesscontinuesmovingunsortedarray boundarybyoneelementtotheright.

Thisalgorithmisnotsuitableforlargedatasetsasitsaverageandworstcasecomplexityare ofO(n2)wherenareno.ofitems.

**Example**:

We take the below depicted array for our example.



Forthefirstpositioninthesortedlist,thewholelist isscannedsequentially.Thefirst positionwhere14isstoredpresently,wesearchthewholelistandfindthat10isthelowest value.



Sowereplace14with10.Afteroneiteration10,whichhappenstobetheminimumvaluein thelist,appearsinthefirstpositionofsortedlist.

Forthesecondposition,where33isresiding,westartscanningtherestofthelistinlinear manner.

Wefindthat14isthesecondlowestvalueinthelistanditshouldappearatthesecond place. We swap thesevalues.



Aftertwoiterations,twoleastvaluesarepositionedatthethebeginninginthesorted manner.

Thesameprocessisappliedontherestoftheitemsinthearray.Weshallseeanpictorial depictionofentiresortingprocess−



Now, we should learn some programming aspects of selection sort.

**Pseudocode**

procedure selection sort list : array of items

n :sizeoflistfori=1ton-1

/\* set current element as minimum\*/

min = i

/\*checktheelementtobeminimum\*/ forj=i+1ton

iflist[j]<list[min]then min =j;

end if end for

/\*swaptheminimumelementwiththecurrentelement\*/ if indexMin != ithen

swaplist[min]andlist[i] end if

endfor

end procedure

**Bubble sort:**

Bubble sort is a simple sorting algorithm. This sorting algorithm is comparison based algorithm in which each pair of adjacent elements is compared and elements are swapped 1if they are not in order .This algorithm is not suitable for large datasets as its average and worst case complexity are of O(n2) where n are no. of items.

How bubble sort works?

Wetakeanunsortedarrayforourexample.BubblesorttakeΟ(n2)times o we'rekeeping short and precise.



Bubblesortstartswithveryfirsttwoelements,comparingthemtocheckwhichoneis greater.



Inthiscase,value33isgreaterthan14,soitisalreadyinsortedlocations.Next,we compare 33 with27.



We find that 27 is smaller than 33 and these two values must be swapped.



Next we compare 33 and 35. We find that both are in already sorted positions.



Then we move to next two values, 35 and 10.



We know than 10 is smaller 35. Hence they are not sorted.



Weswapthesevalues.Wefindthatwereachattheendofthearray.Afteroneiterationthe array should look like this−



Tobeprecise,wearenowshowingthathowarrayshouldlooklikeaftereachiteration. Afterseconditeration,itshouldlooklikethis−

Notice that after each iteration, at least one value moves at the end.



And when there's no swap required, bubble sorts learns that array is completely sorted.



Now we should look into some practical aspects of bubble sort.

**Pseudocode**

Weassume**list**isanarrayof**n**elements.Wefurtherassumethat**swap**function,swapsthe valuesofgivenarrayelements.

begin BubbleSort(list) forallelementsoflist

if list[i] >list[i+1] swap(list[i], list[i+1])

end if end for return list

end BubbleSort

WeobserveinalgorithmthatBubbleSortcompareseachpairofarrayelementunlessthe wholearrayiscompletelysortedascending.Thismaycausefewcomplexityissueslikewhat ifthearrayneedsnomoreswappingasalltheelementsarealreadyascending.

Toease-outtheissue,weuseoneflagvariable**swapped**whichwillhelpustoseeifany swapishappenedornot.Ifnoswapisoccurred,i.e.thearrayrequiresnomoreprocessingto besorted, itwillcomeoutoftheloop.

**CONCLUSION:**Thuswehaveimplementedpythonprogramforsortingelementsusing Selectionsort&Bubblesortmethods.

# Experiment No. 5

**Title:Sortingof anarrayusing**insertionandshell sort.

* TostoreNnumberelementsinarray/list
* Toapplyspecificstrategyforsorting
* ToperformInsertionandShellsortonarray

**Problem Statement:** - Write a Python program to store first year percentage of students in array. Writefunctionfor sortingarrayoffloatingpointnumbersinascendingorderusing

1. InsertionSort
2. Shellsort anddisplaytopfivescores

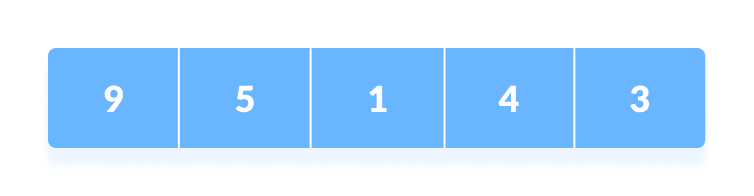
**Outcome:**

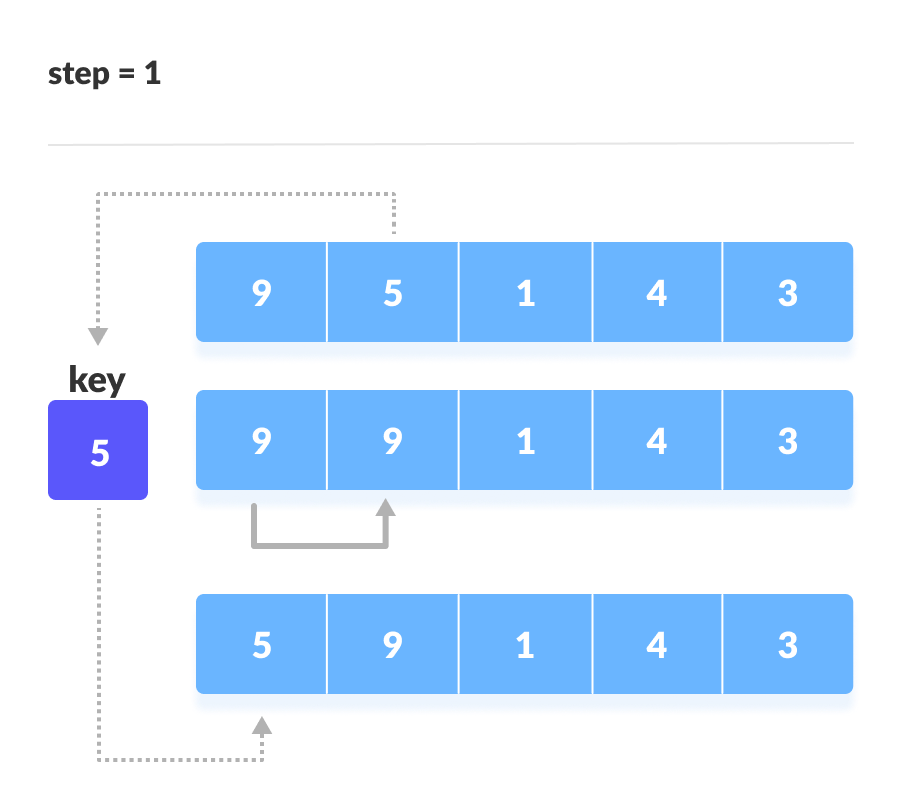
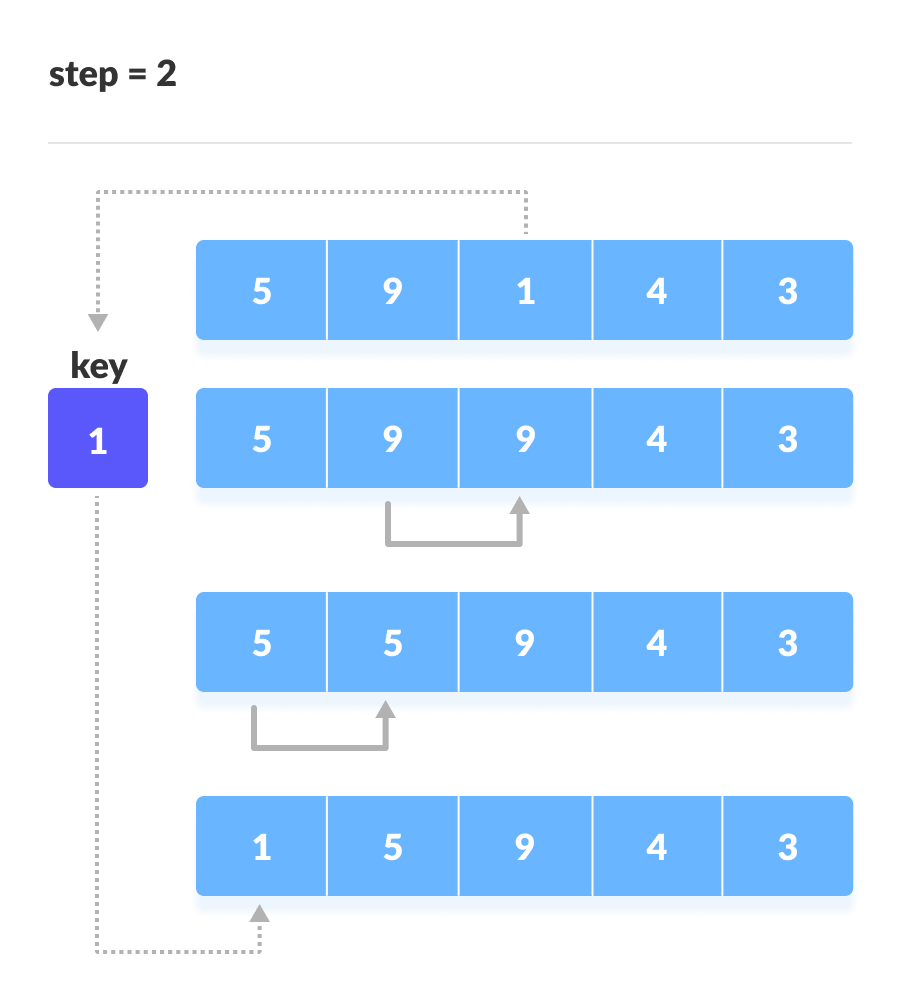
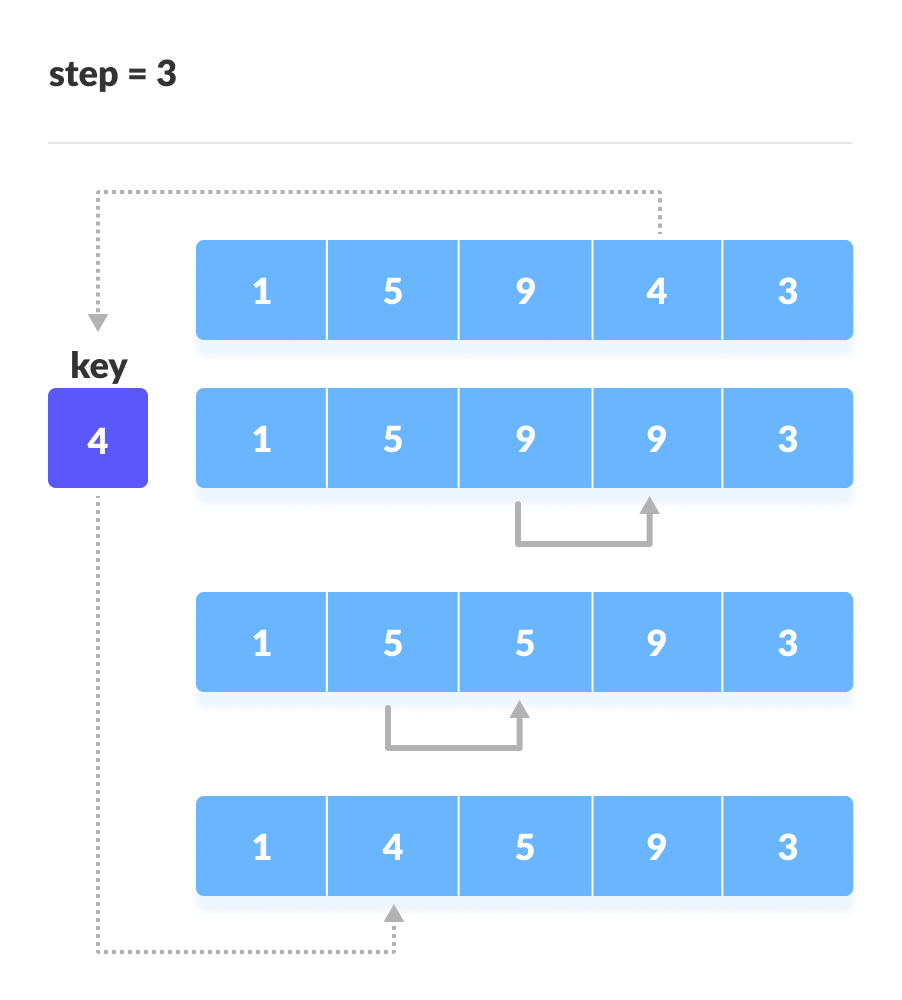
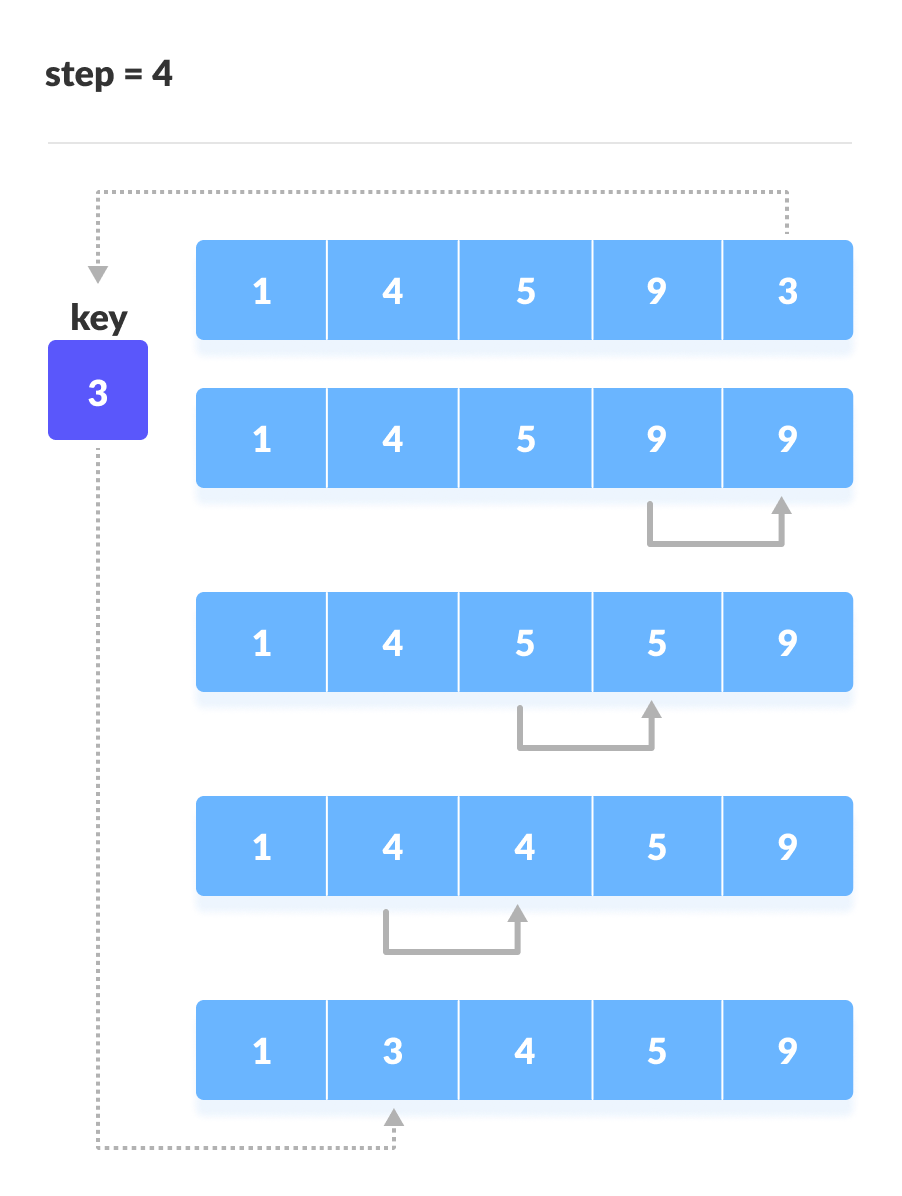
* DisplayUnsortedarray/elementsacceptedbyuser
* PerformInsertion&Shellsortanddisplaysortedelementsandtopfivescores

a**) Insertion Sort Algorithm:**Insertion sort is [a sorting algorithm](https://www.programiz.com/dsa/sorting-algorithm) that places an unsorted element at its suitable place in each iteration.Insertion sort works similarly as we sort cards in our hand in a card game.

## Working of Insertion Sort

Suppose we need to sort the following array.

Initial array

1. The first element in the array is assumed to be sorted. Take the second element and store it separately in key.  
     
   Compare key with the first element. If the first element is greater than key, then key is placed in front of the first element.If the first element is greater than key, then key is placed in front of the first element.
2. Now, the first two elements are sorted.  
     
   Take the third element and compare it with the elements on the left of it. Placed it just behind the element smaller than it. If there is no element smaller than it, then place it at the beginning of the array.Place 1 at the beginning
3. Similarly, place every unsorted element at its correct position.Place 4 behind 1Place 3 behind 1 and the array is sorted

## Insertion Sort Algorithm

insertionSort(array)

markfirst element as sorted

foreach unsorted element X

'extract'the element X

forj <- lastSortedIndex down to 0

ifcurrent element j > X

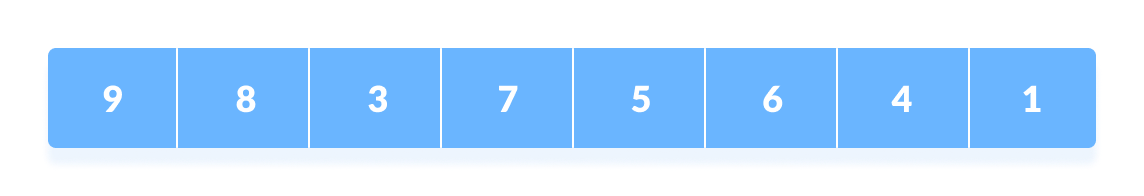
movesorted element to the right by 1

breakloop and insert X here

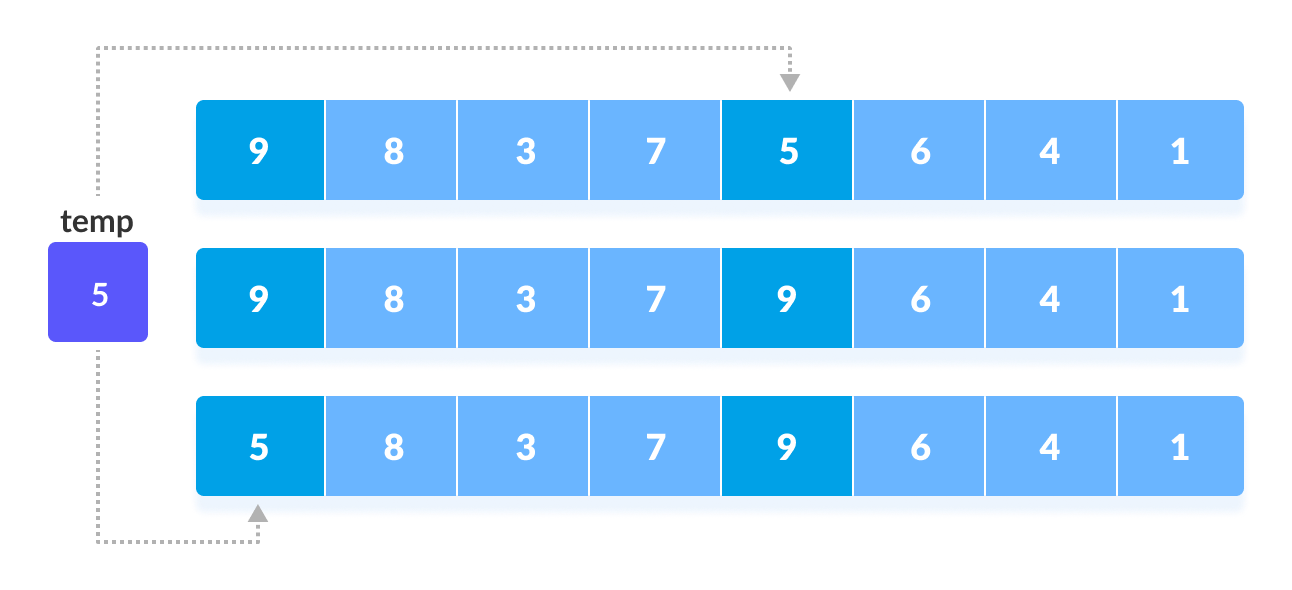
endinsertionSort

# b) Shell Sort Algorithm:Shell sort is a generalized version of the [insertion sort algorithm](https://www.programiz.com/dsa/insertion-sort). It first sorts elements that are far apart from each other and successively reduces the interval between the elements to be sorted.

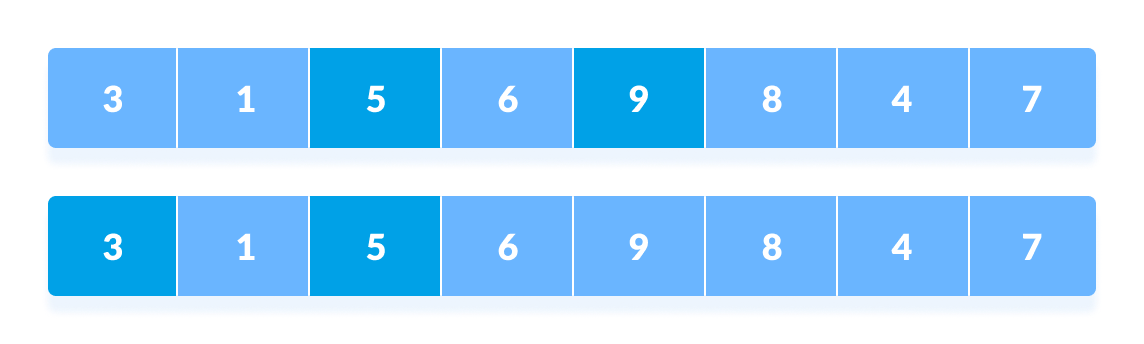
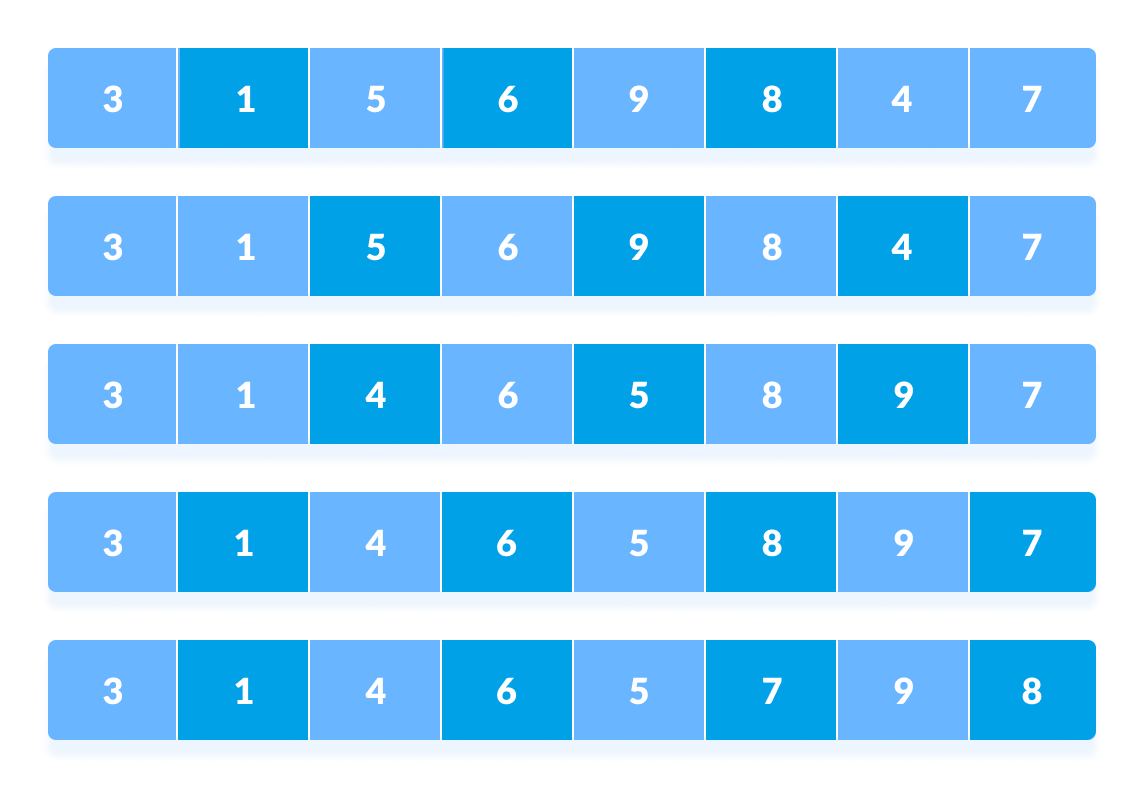
## Working of Shell Sort:

1. Suppose, we need to sort the following array.

Initial array

1. We are using the shell's original sequence (N/2, N/4, ...1) as intervals in our algorithm.  
     
   In the first loop, if the array size is N = 8 then, the elements lying at the interval of N/2 = 4 are compared and swapped if they are not in order.
   1. The 0th element is compared with the 4th element.
   2. If the 0th element is greater than the 4th one then, the 4th element is first stored in temp variable and the 0th element (ie. greater element) is stored in the 4th position and the element stored in temp is stored in the 0th position.Rearrange the elements at n/2 interval  
      This process goes on for all the remaining elements.

Rearrange all the elements at n/2 interval

1. In the second loop, an interval of N/4 = 8/4 = 2 is taken and again the elements lying at these intervals are sorted.Rearrange the elements at n/4 interval  
   You might get confused at this point.All the elements in the array lying at the current interval are compared.  
   The elements at 4th and 2nd position are compared. The elements at 2nd and 0th position are also compared. All the elements in the array lying at the current interval are compared.
2. The same process goes on for remaining elements.Rearrange all the elements at n/4 interval
3. Finally, when the interval is N/8 = 8/8 =1 then the array elements lying at the interval of 1 are sorted. The array is now completely sorted.
4. Rearrange the elements at n/8 interval

## Shell Sort Algorithm:

shellSort(array, size)

for interval i <- size/2n down to 1

for each interval "i" in array

sort all the elements at interval "i"

end shellSort

**Writealgorithm/pseudocode foreach function:**

* 1. Toacceptpercentagefromuser forNnumberofstudents.
  2. ToperformInsertionsortandprintsortedelements
  3. ToperformShell sortandprintsortedelements
  4. Todisplaytop fivescores ofInsertionsort
  5. To displaytop fivescoresof Shellsort

# Experiment No 6

AIM:ImplementationofQuickSort. OBJECTIVES:

* Tounderstandtheconceptofsortingmechanism.
* Toexplorethedeepconceptoftimecomplexitiesofdifferentsortingtechniques.

PROBLEM STATEMENT:

WriteC++programtostorefirstyearpercentageofstudentsinarray.Sortarrayoffloating pointnumbersinascendingorderusingquicksortanddisplaytopfivescores.

OUTCOMES:

 Toanalyze&understandvarioussearching&sortingalgorithm.

THEORY:

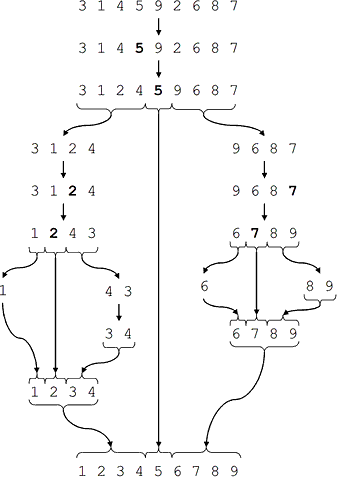
**Quick Sort:**

As one of the more advanced sorting algorithms, you might think that the Quicksort Algorithmissteepedincomplicatedtheoreticalbackground,butthisisnotso.LikeInsertion Sort,thisalgorithmhasafairlysimpleconceptatthecore,butismadecomplicatedbythe constraintsofthearraystructure.

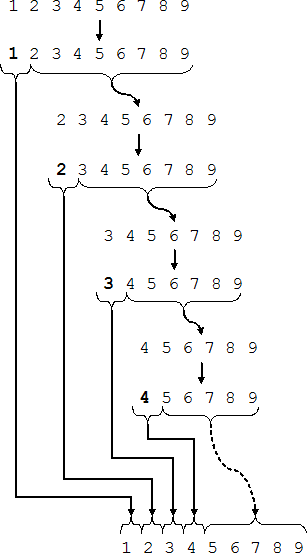
Thebasicconceptistopickoneoftheelementsinthearrayasapivotvaluearoundwhich theotherelementswillberearranged.Everythinglessthanthepivotismovedleftofthe pivot(intotheleftpartition).Similarly,everythinggreaterthanthepivotgoesintotheright partition.Atthispointeachpartitionisrecursivelyquicksorted.

TheQuicksortalgorithmisfastestwhenthemedianofthearrayischosenasthepivotvalue. Thatisbecausetheresultingpartitionsareofverysimilarsize.Eachpartitionsplitsitselfin twoandthusthebasecaseisreachedveryquickly.

Inpractice,theQuicksortalgorithmbecomesveryslowwhenthearraypassedtoitisalready closetobeingsorted.Becausethereisnoefficientwayforthecomputertofindthemedian elementtouseasthepivot,thefirstelementofthearrayisusedasthepivot.Sowhenthe arrayisalmostsorted,Quicksortdoesn'tpartitionitequally.Instead,thepartitionsare lopsidedlikeinFigure2.Thismeansthatoneoftherecursionbranchesismuchdeeperthantheother,andcausesexecutiontimetogoup.Thus,itissaidthatthemorerandomthe arrangementofthearray,thefastertheQuicksortAlgorithmfinishes.



**Figure :** The ideal Quicksort on a random array.



**Figure :** Quicksort on an already sorted array.

ThesearethestepstakentosortanarrayusingQuickSort. Thepseudocodefortheabovealgorithmcanbederivedas−

function partitionFunc(left, right, pivot) leftPointer = left -1

rightPointer = right while Truedo

while A[++leftPointer] <pivotdo

//do-nothing end while

while rightPointer >0 &&A[--rightPointer] >pivot do

//do-nothing end while

if leftPointer >= rightPointer

break

else

swap leftPointer,rightPointer end if

end while

swap leftPointer,right return leftPointer

end function

To get more into it, let see the pseudocode for quick sort algorithm −

procedure quickSort(left, right) if right-left <=0

return else

pivot = A[right]

partition=partitionFunc(left,right,pivot) quickSort(left,partition-1) quickSort(partition+1,right)

end if

end procedure

**CONCLUSION:**ThuswehaveimplementedpythonprogramtoimplementQuicksortto display top 5scores

# Experiment No: 7

**Title:**

Write C++ program for storing binary number using doubly linked lists. Write functions

a) to compute 1‘s and 2‘s complement

1. b) add two binary numbers

**Index terms:** Binary Numbers, Doubly linked list

**Theory:**

* **Doubly Linked list:**

A **D**oubly **L**inked **L**ist (DLL) contains an extra pointer, typically called previous pointer, together with next pointer and data which are there in singly linked list.



**Node structure of DLL in C :**

/\* Node of a doubly linked list \*/

structnode

{

   intdata;

   structnode \*next; // Pointer to next node in DLL

   structnode \*prev; // Pointer to previous node in DLL

};

# 1’s and 2’s complement of a Binary Number

**1’s complement** of a binary number is another binary number obtained by toggling all bits in it, i.e., transforming the 0 bit to 1 and the 1 bit to 0.

Examples:

1's complement of "0111" is "1000"

1's complement of "1100" is "0011"

**2’s complement** of a binary number is 1 added to the 1’s complement of the binary number.  
Examples:

2's complement of "0111" is "1001"

2's complement of "1100" is "0100"

**Algorithm:**

1. Read decimal value from user.
2. Convert it into binary format.
3. Store all digits in Doubly linked list.
4. Finds the 1's complement of the binary number

for(i=0; i<SIZE; i++)

{

if(binary[i]=='1')

{ onesComp[i] = '0';

} else if(binary[i]=='0')

{ onesComp[i] = '1';

}

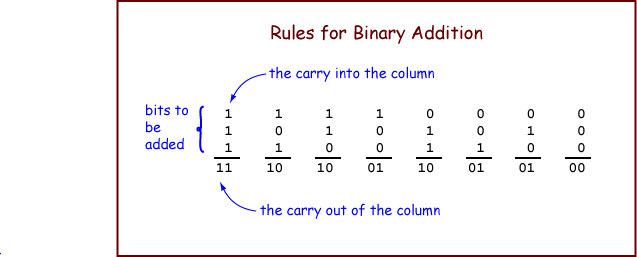
}

onesComp[SIZE] = '\0';

1. Find 2’ s complement by adding one to 1's complement.
2. Print 1's complement and 2s complement.

* **Addition of binary numbers:**

**Rules for binary addition:**



**Conclusion**Thus I have studied program for storing binary number using doubly linked lists. Write functions

a) to compute 1‘s and 2‘s complement

b) add two binary numbers

# Experiment No 8

**Title:**setoperationsusinglinkedlist.

**Objectives:**To understand set operation. Representation andimplementation of operation of sets using linked list

## ProblemStatement:

Second year Computer Engineering class, set A of students like Vanilla Ice- cream and set B of students like butterscotch ice-cream. Write C++ programto store two sets using linked list. Compute and display-

1. Setofstudentswholikebothvanillaandbutterscotch
2. Setofstudentswholikeeithervanillaorbutterscotchornotboth
3. Numberofstudentswholike neither vanillanorbutterscotch

## Outcomes:

1. Listofstudentswholikeeithervanillaorbutterscotchorboth
2. List ofstudentswholikebothvanillaandbutterscotch
3. Listofstudentswholikeonly vanillanot butterscotch
4. Listofstudentswho likeonlybutterscotchnotvanilla
5. Numberofstudentswholikeneithervanillanorbutterscotch

## Software&Hardwarerequirements:

LinuxoperatingsystemEclipseIDEwith g++compiler

## Theory-Conceptinbrief:

Set:

a set is a collection ofobjects which are called the members or elements of thatset.Ifwehaveasetwesaythatsomeobjectsbelong(ordonotbelong)tothis set, are (or are not) in the set.

Examples:thesetofstudentsinthisroom;theEnglishalphabetmaybe viewed as the set of letters of the English language; the set of natural numbers Operations on sets:

1. Union: The union of A and B, written A 𝖴 B , is the set whose elements are just the elements of A or B or of both.
2. Intersection: The intersection of A and B , written A ∩ B , is the set whoseelements are just the elements of both A and B.
3. Difference:Anotherbinaryoperationonarbitrarysetsisthedifference― A minusB‖,writtenA–B,which‗subtracts fromAallelementswhicharein

B.[Alsocalledrelativecomplement:thecomplementofBrelativetoA.]

1. Complement: This operation is creating a set A, which is the setconsisting of everything not in A

## Algorithm:

1. **Algorithm difference(struct node \*head1, struct node \*head2) Precondition:**Accept the two sets of name in the form of linked list.**Post condition**: Difference of two sets

**Return:**headnodeoftheresultantlinkedlist(Difference)

* 1. Setp=head1,i.e.headof1stlinkedlist
  2. Initializethe 3rdlinked listasemptyi.e.head3=NULL
  3. while(p!=NULL)
     1. Setflag=0
     2. Setq=head2,i.e.headof2ndlinkedlist
     3. while(q!=NULL)
        1. if(strcmp(p->name,q->name)==0)
           1. Setflagto1andgotostepd
        2. else
           1. Moveqtothenextnode,q=q->next
     4. if(flag!=1)
        1. if(head3==NULL)
           1. AllocatememoryforNewnode
           2. strcpy(New->name,p->name)
           3. Setr=Newandr->next=NULL
           4. MakeNewnodeasheadnodeofresultantlinkedlist
        2. else
           1. AllocatememoryforNewnode
           2. r->next=New
           3. strcpy(New->name,p->name)
           4. Setr=r->nextandr->next=NULL
     5. Moveptothenextnode,p=p->next
  4. returnhead3i.e.headofresultantlinkedlist
  5. Stop

**Algorithm Intersection(struct node \*head1, struct node \*head2) Precondition:**Acceptthetwosetsofnameintheformoflinkedlist. **Post condition**: Intersection of two sets

**Return:**Nil

1. Setp=head1,i.e.headof1stlinkedlist
2. Initialize the 3rdlinked listasemptyi.e.head3=NULL
3. while(p!=NULL)
   1. Setq=head2,i.e.headof2ndlinkedlist
   2. while(q!=NULL)
   3. f(strcmp(p->name,q->name)==0)
      1. if(head3==NULL)
         1. AllocatememoryforNewnode
         2. strcpy(New->name,p->name)
         3. Setr=Newandr->next=NULL;
         4. Makenew node asheadof linkedlist
      2. else
         1. AllocatememoryforNewnode
         2. r->next=New
         3. strcpy(New->name,p->name)
         4. Setr=r->nextandr->next=NULL
      3. Gotostepc

**ii.**Moveqtothenextnode,q=q->next

* 1. Moveptothenextnode,p=p->next

1. Call display(head3) function to display resultant linked list containing intersection
2. Stop
3. **Algorithm Union(struct node \*head1,struct node \*head2) Precondition:**Acceptthetwosetsofnameintheformoflinkedlist. **Post condition**: Union of two sets

**Return:**headnodeoftheresultantlinkedlist(Union)

* 1. Setq=head2,i.e.headof2ndlinkedlist
  2. Copyallcontentsoffirstlinkedlistinthird,head3=head1
  3. Setr=head3,i.e.headof3rdlinkedlist
  4. while(r->next!=NULL)Movertothenextnode,r=r->next
  5. while(q!=NULL)
     1. Setp=head1,i.e.headof1stlinkedlist
     2. Initializeflagto0
     3. while(p!=NULL)
        1. if(strcmp(p->name,q->name)==0)Setflagto1andgotostepd
        2. elseMoveptothenextnode,p=p->next
     4. if(flag==0)
        1. AllocatememoryforNewnode
        2. r->next=New
        3. strcpy(New->name,q->name);
        4. Setr=r->nextandr->next=NULL
     5. Moveqtothenextnode,q=q->next
  6. returnhead3,i.e.headofresultantlinkedlistcontainingunion
  7. Stop

**Conclusion:**

ThusIhavestudiedconceptofsetanditsrepresentationusingarray.Ihavealso implemented all the operations of set.

# Experiment No. 9

**AIM:** Write a C++ program using stack to check whether given expression is well parenthesized or not.

AIM:Tocheckthegivenexpressioniswellparenthesizedornot. OBJECTIVES:

* Tounderstandtheconceptofstack.
* Toexplorethedeepconceptofpointersandparenthesis.
* Tounderstandthedifferentoperationsonstack.
* Tounderstandthememoryrequirementforstackdatastructures.

PROBLEM STATEMENT:

In any language program mostly syntax error occurs due to unbalancing delimiter such as (),{},[]. Write C++ program using stack to check whether given expression is well parenthesized or not.

OUTCOMES:

 Todesignalltheaspectsofstack&itsvariants.

SOFTWARE & HARDWARE REQUIREMENTS:

Softwares: Eclipse/Geany Editor/gedit

G++ compiler

Fedora20OperatingSystem Hardware:PentiumDualCore(3.00GHz)

2 GB RAM

THEORY:

*Basic operations onStack*

1. **voidinitialize(stack\*P)**:Itinitializesastackasanemptystack.Initialvalueof stack is set to-1.

void initialize (stack \*P)

{

P -> top = -1;

}

1. **intempty(stack\*P)**:Functioncheckswhetherthestackisempty.Itreturns1or0 dependingonwhetherthestackisemptyornot.

int empty ( stack \*P)

{

if(P->top==-1)

return(1);

return (0);

}

1. **intfull(stack\*P)**:Functioncheckswhetherthestackisfull.Wheneverthestackis full, top points to the last element (ie. MAX-1) of the array. It returns 1 or 0 dependingonwhetherthestackisfullornot.

int empty ( stack \*P)

{

if (P -> top ==)

return (1);

return (0);

}

1. intpush(stack\*P,intx):Thefunctioninsertstheelementxontothestackpointedby

P.Insertionwillcauseanoverflowifthestackisfull. voidpush(stack\*P,intx)

{

P->top=P->top+1; P->data[P->top]=x;

}

1. intpop(stack\*P):Thefunctiondeletestopmostelementfromthestackandalso returnsittothecallingprogram.Deletionfromanemptystackwillcauseunderflow.

int pop ( stack \*P)

{

int x;

x=P->data[P->top]; P->top=P->top-1; return(x);

}

*Applications ofStack*

1. ExpressionConversion
2. ExpressionEvaluation
3. Parsing well-formedparenthesis
4. Recursion
5. FunctionCall
6. Decimal to binaryconversion
7. Reversing astring.

##### Well formedness ofparenthesis

Anexpressionissaidtobewellformedifeveryopeningbrackethasacorrespondingclosing bracket and there is no extrabracket.

Examples (a + b- (c +d))

(p–(q/(r+s)^(t–u))\*v) (a–[b/c\*{d+e/f}^g])

Algorithm:

* Declare a character stackS.
* Nowtraversetheexpressionstringexp.
  1. Ifthecurrentcharacterisastartingbracket(**‘(‘or‘{‘or‘[‘**)thenpushittostack.
  2. Ifthecurrentcharacterisaclosingbracket(**‘)’or‘}’or‘]’**)thenpopfromstack andifthepoppedcharacteristhematchingstartingbracketthenfineelsebrackets are notbalanced.
* Aftercompletetraversal,ifthereissomestartingbracketleftinstackthen“not balanced”

**CONCLUSION:**ThusweimplementedC++programtocheckwhetherthegivenexpression is well parenthesized ornot.

# Assignment No 10

**Title:** Infix to Postfix Conversion and valuation

**Objective:**

1. Understandtheconceptofhowtoconvertinfixtopostfix

expression.

1. Understandhowtoevaluatetheexpressionusingstack**.**

## ProblemStatement:

ImplementC++programforexpressionconversionasinfixtopostfixandits evaluation using stack based on given conditions:

1. Operandsandoperator,bothmustbesinglecharacter.
2. InputPostfix expressionmust beinadesiredformat.
3. Only'+','-','\*'and'/'operatorsareexpected.

**Outcome:**

1. Will be able to understand theconcept of how to convert infixtopostfix expression.
2. Will be able to understand how to evaluate the expressionusing stack.

**Software&HardwareRequirements:**

1. 64-bitOpensourceLinuxoritsderivative
2. OpenSourceC++ProgrammingtoollikeG++/GCC

## Theory:

**Infixexpression:**Theexpressionoftheformaopb.Whenanoperatorisin- between every pair of operands.

**Postfixexpression:**Theexpressionoftheformabop.Whenanoperatoris followed for every pair of operands.

## Needofpostfixrepresentationoftheexpression

The compiler scans the expression either from left to right or fromright to left. Considerthebelowexpression:aop1bop2cop3dIfop1=+,op2=\*,op3=+

Thecompilerfirstscanstheexpressiontoevaluatetheexpressionb\*c,thenagainscantheexpressiontoaddatoit.Theresultisthen added to d after another scan. The repeated scanning makesit very in- efficient. It is better to convert the expression to postfix (or prefix) form before evaluation.

The corresponding expression in postfix form is: abc\*+d+. The postfix expressions can be evaluated easily using a stackToimplementconversionofaninfixexpressionintopostfixexpression.

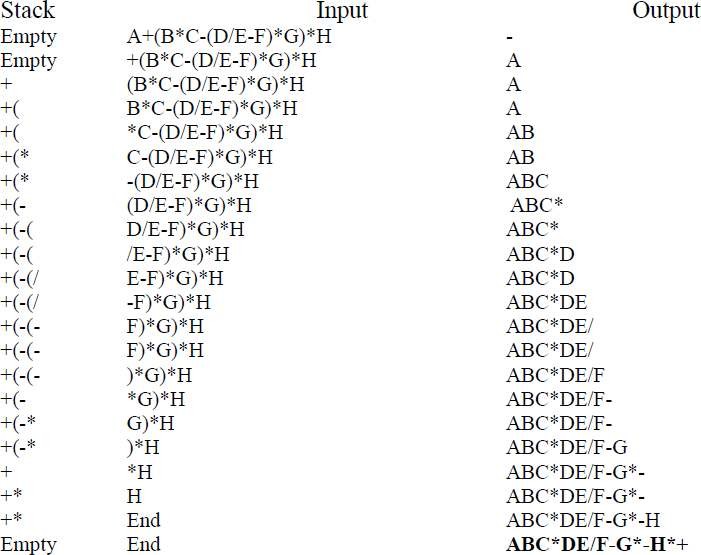
Algorithm:

1. Defineastackoftoholdthecharacters.
2. Initializestacktop=-1.
3. Readtheinfix expression.
4. Add„)‟totheendoftheinfixexpression.
5. Push„(„tothestack.
6. Scan the infixexpression from left torightand repeat the step7 for all the characters in the infix expression.
7. If the character is an operand Add it to the postfix expression Ifthe character is „(„ Push it to the stack If the character is „)‟ Repeatedly Popthecharacters fromthestackand addit tothe postfixexpression until „)‟ is encountered. Pop „)‟ If the characteris an operatorIf theprecedence ofthe character is lesserthan orequal to the precedence of the operator in the top of the stack repeatedly pop the characters from the stack and add it to the post fix expression till anoperatorofhigherprecedenceis encountered.Pushtheoperator to the top of the stack.
8. Printthepostfixexpression.
9. Stop.

Conversion To Postfix

EXAMPLE:

A+(B\*C-(D/E-F)\*G)\*H



Toimplement evaluation of apostfix expression. AlgorithmforPostfixEvaluation:

1. Declarethestructureforthestack.
2. Readthepostfixexpression.
3. Repeatedly execute the following for all the characters in the expression from left to right If the character is a number Then convert the character to integer by subtracting „0‟ from thecharacter. Push the integer value into the stack. If the characteris a operator Pop two values from the stack and perform the operation and store the result to stack.
4. Thestacktophastheresult.PopitandPrint.

1. Stop

## Flowchart:

**DrawFlowchart**

## TestCases:

Input:

InfixExpression:(3\*2)/(5-3)Output:

PostfixExpression:32\*53-/

Evaluationresult:3

## Conclusion/Analysis:

Understoodtheconceptofhowtoconvertinfixtopostfixexpressionandhowto evaluatetheexpressionusingstack**.**

Experiment No11 **AIM:Implementationofjobqueueforoperatingsystem OBJECTIVES:**

* + Tounderstandtheconceptofqueue.
  + Toexplorethedeepconceptinsertion&deletionofelementsin/fromqueue.

PROBLEM STATEMENT:

Queuesarefrequentlyusedincomputerprogramming,andatypicalexampleisthecreationofa jobqueuebyanoperatingsystem.Iftheoperatingsystemdoesnotusepriorities,thenthejobsare processedintheordertheyenterthesystem.WriteC++programforsimulatingjobqueue.Write functionstoaddjobanddeletejobfromqueue.

OUTCOMES:

 Torealizedifferenttypesofqueue&itsimplementation.

Software & Hardware requirements:

Softwares: Eclipse/Geany Editor/Gedit

GCC & G++ compiler Fedora20OperatingSystem

Hardware: Pentium Dual Core (3.00GHz)

2 GB RAM

THEORY:

Queueisanabstractdatastructure,somewhatsimilartoStack.IncontrasttoQueue,queue isopenedatbothend.Oneendisalwaysusedtoinsertdata(enqueue)andtheotherisused toremovedata(dequeue).QueuefollowsFirst-In-First-Outmethodology,i.e.,thedataitem stored first will be accessedfirst.

Arealworldexampleofqueuecanbeasingle-laneone-wayroad,wherethevehicleenters first,exitsfirst.Morereal-worldexamplecanbeseenasqueuesatticketwindows&bus- stops.

**Queue Representation**

Aswenowunderstandthatinqueue,weaccessbothendsfordifferentreasons,adiagram givenbelowtriestoexplainqueuerepresentationasdatastructure−



SameasQueue,queuecanalsobeimplementedusingArray,Linked-list,Pointerand Structures.Forthesakeofsimplicityweshallimplementqueueusingone-dimensional array.

**Basic Operations**

Queue operations may involve initializing or defining the queue, utilizing it and then completingerasingitfrommemory.Hereweshalltrytounderstandbasicoperations associated with queues−

* **enqueue()** − add (store) an item to thequeue.
* **dequeue()**−remove(access)anitemfromthequeue.

Fewmorefunctionsarerequiredtomakeabovementionedqueueoperationefficient.These are−

* **isfull()**−checksifqueueisfull.
* **isempty()** − checks if queue isempty.

Inqueue,wealways**delete**(oraccess)data,pointedby**front**pointerandwhileinserting(or storing)datainqueuewetakehelpof**rear**pointer.

Let's first learn about supportive functions of a queue –

isfull()

Asweareusingsingledimensionarraytoimplementqueue,wejustcheckfortherear pointertoreachatMAXSIZEtodeterminethat queueisfull.Incasewemaintainqueueina circularlinked-list,thealgorithmwilldiffer.Algorithmofisfull()function−

begin procedure isfull

if**rear**equalstoMAXSIZE returntrue

else

return false endif

end procedure

Implementation of isfull() function in C programming language −

bool isfull() {

if(rear == MAXSIZE - 1) return true;

else

return false;

}

isempty()

Algorithmofisempty()function−

begin procedure isempty

if **front** is less than MIN OR **front** is greater than **rear**

return true else

return false endif

end procedure

Ifvalueof**front**islessthanMINor0,ittellsthatqueueisnotyetinitialized,henceempty. Here'stheCprogrammingcode−

bool isempty() {

if(front <0 || front >rear)

return true; else

return false;

}

**Insertion Operation:**

Asqueuemaintainstwodatapointers,**front**and**rear**,itsoperationsarecomparativelymore difficult to implement thanQueue.

The following steps should be taken to insert data into a queue −

* **Step1**−Checkifqueueisfull.
* **Step2**−Ifqueueisfull,produceoverflowerrorandexit.
* **Step3**−Ifqueueisnotfull,increment**rear**pointertopointnextemptyspace.
* **Step4**−Adddataelementtothequeuelocation,whererearispointing.
* **Step 5** − returnsuccess.

Sometimes,wealsocheckthatifqueueisinitializedornottohandleanyunforeseen situations.

**Deletion Operation:**

Accessingdatafromqueueisaprocessoftwotasks−accessthedatawhere**front**is pointing and remove the data after access. The following steps are taken to perform **delete**operation −

* **Step1**−Checkifqueueisempty.
* **Step2**−Ifqueueisempty,produceunderflowerrorandexit.
* **Step3**−Ifqueueisnotempty,accessdatawhere**front**ispointing.
* **Step4**−Increment**front**pointertopointnextavailabledataelement.
* **Step 5** − returnsuccess.

**CONCLUSION:**ThuswehaveimplementedC++programforjobqueueforoperating system.

# Experiment No: 13

**Title:**PerformthedifferentoperationsonCircularQueue.

## Objectives:

1. UnderstandthebasicconceptofCircularQueueImplem
2. entation.
3. Understandtheconceptofinsertion,deletioninacircularqueue.

**Problem Statement:** Pizza parlor accepting maximum M orders. Orders are served in first come first served basis. Order once placed cannot be cancelled. Write C++ program to simulatethe system using circular queueusing array.

**Outcomes:**Uponcompletionofassignment,studentwillbeableto:

* Willbeabletounderstand theconceptofinsertionanddeletionin circularqueue **Software Requirements:**64-bitOpen sourceLinuxor its derivative,Windows 8/9/10,ProgrammingTools: G++/GCCcompiler,EclipseIDE.

**HardwareRequirements:**Intel ®CoreTMi3-3220CPU@3.30GHz,4GBRAM,500GB ATA HDD

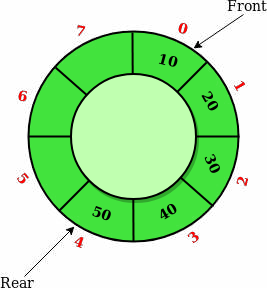
**DateofCompletion:**

**AssessmentGrade/Marks:**

**Assessor’sSign:**

**TheoryConceptinBrief:**

Circular Queue is a linear data structurein which the operations areperformed based on FIFO (FirstInFirstOut)principleandthelastpositionis connectedbacktothefirstpositiontomakea circle.It is alsocalled **‘Ring Buffer’**.



## BasicOperations

Someofthebasic operationsofthecircularqueueareasfollows:

1. **Front:**Returnsthefrontpositioninthecircularqueue.
2. **Rear:**Returnstherearpositioninthecircularqueue.
3. **Enqueue:**Enqueue (value) is usedtoinsert anelementin the circularqueue. The element is always insertedat the rearend of the queue.

Wefollowthefollowingsequenceofstepstoinsertanewelementinthecircular queue.

#1)Checkif the circular queue is full:test ((rear== SIZE-1 &&front ==0) ||(rear== front- 1)), where „SIZE‟is the size of the circularqueue.

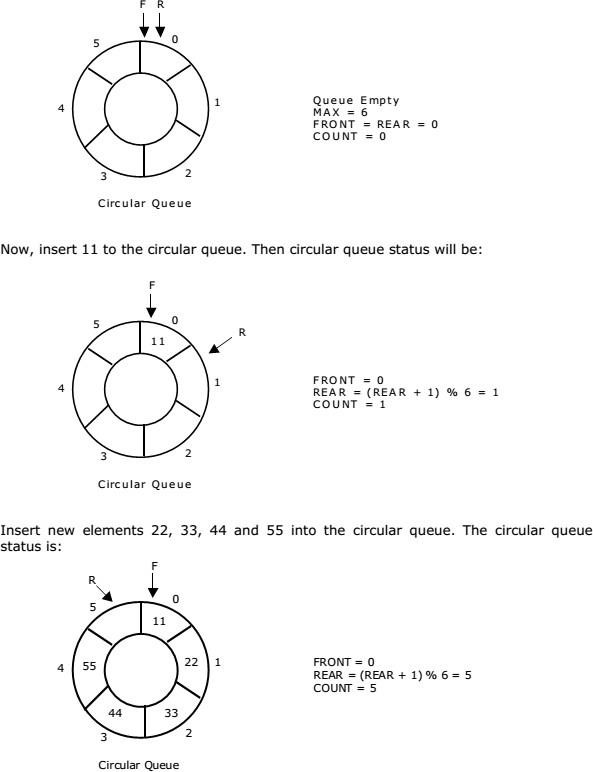
#2) Ifthe circular queueis fullthen it displays amessage as“Queue is full”.If queue is not fullthen, check if(rear == SIZE – 1 && front != 0). If it is true then set rear=0 and insert element.

1. **Dequeue:** Dequeue function is used to delete an element from the queue. In the circular queue, the element is always deleted from the front end. Given below is the sequence for dequeue operation in a circular queue.

## Steps:

1. Checkifthecircularqueueis Empty:checkif(front==-1).
2. Ifitisempty thendisplaythemessage“Queueis empty”. Ifqueueisnot emptythen perform step 3.
3. Checkif(front==rear).Ifitistruethensetfront=rear=-1elsecheckif(front==size-1), ifitis

truethensetfront=0andreturntheelement.



## RepresentationofCircularQueue:Letusconsideracircularqueue,whichcanhold maximum (MAX) of six elements. Initially queue is empty.

## Application ofCircularQueue:

* 1. **CPU Scheduling:** Operating system process that requires some event to occur or for someother processesto complete for execution is often maintained in a circular queue so that they execute one after the other when all the conditions are met or when all events occur.
  2. **Memory Management:**Use of ordinary queues wastes memory space as already mentioned in our abovediscussion.Using a circular queue formemory management is beneficial for optimummemory usage.
  3. **Computer Controlled Traffic Signal System:** Computerized traffic signals are often added to a circular queueso that they repeat themselvesafter the specified time interval has elapsed.

## DrawbackofCircular Queue

The drawbackofcircularqueue is , difficult todistinguishedthe fullandempty cases.Itisalso known as

## boundarycaseproblem.

1. Incircularqueueit isnecessarythat:
2. Beforeinsertion,fullnessofQueuemustbechecked(foroverflow).
3. Beforedeletion,emptinessofQueuemustbechecked(forunderflow).

## CircularQueueComplexityAnalysis:

Thecomplexity oftheenqueueanddequeueoperationsof a circularqueueisO(1) for(array implementations).

## AlgorithmofaCircularqueue:

1. Initializethequeue,withsizeofthequeuedefined(maxSize),andheadandtailpointers.
2. enqueue: Check if the number of elements is equal to maxSize - 1:If Yes, then return Queue is full.

IfNo,thenadd thenewdataelement to thelocationoftail pointer and increment the tail pointer.

1. dequeue: Check if the number of elements in the queue is zero:If Yes, then return Queue is empty.

IfNo,thenincrementtheheadpointer.

1. Findingthesize:

If,tail>=head,size=(tail-head)+1

Butif,head>tail,thensize=maxSize-(head-tail)+1

## Flowchart:

DrawFlowchartforaboveoperations.

## Conclusion/Analysis

Inthisway,weunderstoodtheconceptofinsertionanddeletionincircularqueue