1. Write a pseudocode to accept principle, rate of interest and time. Calculate simple interest and display the same

```
Start
        principle ← 0.0 (double);

        time ← 0.0 (double);

        rateInterest ← 0.0 (double);

        simpleInterest← 0.0 (double);


        Accept principle, time, rateInterest;


        simpleInterest ← (principle * time * rateInterest) /100;


        Display simpleInterest;

Stop
```

2. Write a pseudocode to accept two numbers. Display the two numbers. Swap the two numbers and display them again.

```
Start
        num1 ← 0 (integer);

        num2 ← 0 (integer);

        temp ← 0 (integer);


        Accept num1, num2;


        Display num1, num2;
```

```
    temp ← num1;
    num1 ← num2;
    num2 ← temp;

    Display num1, num2;
Stop

Start
    num1 ← 0 (integer);
    num2 ← 0 (integer);

    Accept num1, num2;

    Display num1, num2;

    num1 ← num1 + num2;
    num2 ← num1 – num2;
    num1 ← num1 – num2;

    Display num1, num2;
Stop
```

3. Write a pseudocode to accept a number and display whether it is an even or odd number

```
Start
        num ← 0 (integer);


        Accept num;


        if(num % 2 == 0) then
                Display “Even number”;
        else
                Display “Odd number”;
        endif;
Stop
```

4. Write a pseudocode to accept a double value. Separate the whole value from the fractional value and store them in two variables. Display the same.

```
Start
        value ← 0.0 (double);
        whole ← 0 (integer);
        fractional ← 0.0 (double);
        Accept value;
        whole ← int(value);
        fractional ← value – whole;


        Display whole, fractional;
Stop
```

5. Write a pseudocode to accept a student's name and scores in three subject. Display the average and total. Display whether the student has secured 1st, 2nd, pass class or has failed. 1st class is for a score of 60 and above, 2nd is for a score of 50 and above, while pass class is for a score of 35 and above. If the score is less than 35, then the student fails.

Start

```
    name ← " " (string);

    score1, score2, score3 ← 0 (integer);

    total ← 0 (integer);

    avg ← 0.0 (double);


    Accept name, score1, score2, score3;


    tot ← score1 + score2 + score3;

    avg ← tot / 3.0;


    if(score1 >= 35 and score2 >= 35

                        and score3 >= 35) then

            if (avg >= 60) then

                display "first class;

            else if (avg >= 50) then

                display "second class;

            else

                display "pass class;

            endif;

            endif;

    else
```

```
        display “student fails”;
    endif;
Stop
```

6. Write a pseudocode to find the largest and second largest of 3 numbers

```
Start
    num1, num2, num3 ← 0 (integer);
    large ← 0 (integer);
    secLarge ← 0 (integer);

    Accept num1, num2, num3;

    if(num1 > num2) then
        large ← num1;
        secLarge ← num2;
    else
        large ← num2;
        secLarge ← num1;
    endif;

    if(num3 > large) then
        secLarge ← large;
        large ← num3;
    else if (num3 > secLarge) then
        secLarge ← num3;
    endif;
    endif;


    Display large, secLarge;

Stop
```

7. Write a pseudocode to accept name, empId, basic, special allowances, percentage of bonus and monthly tax saving investments. The gross monthly salary is basic + special allowances. Compute the annual salary. The gross annual salary also includes the bonus. Compute the annual net salary, by deducting taxes as suggested.

Income upto 1 lac – exempted

Income from 1 to 1.5 lac - 20%

Income from 1.5 lac onwards - 30%

However if there is any tax saving investment, then there is further exemption of upto 1 lac annually. This would mean that by having tax saving investments of about 1 lac, an income of 2 lacs is non-taxable. Display the annual gross, annual net and tax payable.

```
Start
    name ← " " (string);

    empId ← " " (string);

    basic ← 0.0 (double);

    splAllow ← 0.0 (double);

    perOfBonus ← 0.0 (double);

    taxSavMnthly ← 0.0 (double);

    grossAnnual, netAnnual, taxPaid ← 0.0 (double);

    mnthlyGross, bonus ← 0.0 (double);



    Accept name, empId, basic, splAllow,

                            percOfBonus, taxSavMnthly;
```

```
mnthlyGross ← basic + splAllow;
bonus ← (basic * 12) * percOfBonus / 100.0;
grossAnnual ← monthlyGross * 12 + bonus;

if(grossAnnual > 100000) then
        if(annTaxSav <= 100000) then
                taxableIncome ←
                        grossAnnual – annTaxSav;
        else
                taxableIncome ←
                        grossAnnual – 100000;
        endif;
        if(taxableIncome > 250000) then
                taxPaid ← 25000 +
                        (taxableIncome -250000) * 0.3;
        else    if(taxableIncome > 150000) then
                taxPaid ← 5000 +
                        (taxableIncome –150000) * 0.2;
        else if (taxableIncome > 100000) then
                taxPaid ←
                        (taxableIncome –100000) * 0.1;
        endif;
        endif;
        endif;
endif;
```

```
    netAnnual ← grossAnnual – taxPaid;

    Display grossAnnual, netAnnual, taxPaid;

Stop
```

8. A vendor offers software services to a client. Each resource is billed at some dollar rate per hour. The total cost of the project for the client is therefore, the total number of hours contributed by all the vendor resources * the dollar rate / hour. There are however some variants.

The vendor might have purchased hardware/infrastructure or software licenses needed for the project.

The vendor might have utilized external consultants for the project.

The client looks at the vendor as a one-stop solution and hence external resources employed by the vendor need to be paid by the vendor.

It might however be possible that the vendor's hardware and software purchases are borne by the client. In this case, the client pays the vendor 30% of the hardware/infrastructure costs. In case of software licenses, the client pays the vendor 50% of the cost, if they are commonly available and used, or 100% if the software is infrequently used or is proprietary client technology.

The external consultants employed by the vendor will come at a dollar rate per hour.

Accept the suitable inputs and display the profits / loss realized by the vendor.

Start

```
    totHrs, ratePerHr ← 0.0 (double);
    hasExternalConsultants ← ‘N’ (char);
    consHrs, consRatePerHr ← 0.0 (double);
    hasHwInfra ← ‘N’ (char);
    hwInfraCosts ← 0.0 (double);
    hasSoftwareLic ← ‘N’ (char);
    swLicCosts ← 0.0 (double);
    freqType ← ‘R’ (char);
    projCost ← 0.0 (double);
    swCost ← 0.0 (double);
    profits ← 0.0 (double);

    Accept totHrs, ratePerHr;

    Accept hasExternalConsultants;

    if(hasExternalConsultants == ‘y’ or
            hasExternalConsultants == ‘Y’) then
            Accept consHrs, consRatePerHr;
    endif;

    Accept hasHwInfra;
    if(hasHwInfra == ‘y’ or hasHwInfra == ‘Y’)
            Accept hwInfraCosts;
```

```
endif;

Accept hasSoftwareLic;
if(hasSoftwareLic == ‘y’ or hasSoftwareLic == ‘Y’)
    Accept swLicCosts;
    Accept freqType;
endif;

projCost ← totHrs * ratePerHr;
projCost ← projCost + (hwInfraCosts) * 0.3;

Note:   C – stands for Common, R – stands for
                Rare

if(freqType == ‘C’) then
    swCost ← 0.5 * swLicCosts;
else if(freqType == ‘R’) then
    swCost ← swLicCosts;
endif;

projCost ← projCost + swCost;

Display projCost;

profits ← projCost –
    ( (consHrs * consRatePerHr) +
                (hwInfraCosts) + (SwLicCosts));


        Display profits;


        If(profits > 0) then
                Display "Profitable";
        else
                Display "Incurs a loss";
        endif;
Stop
```

9. Write a pseudocode to find the sum of all odd numbers from 1 to N. Accept N. Display the sum.

```
Start
        i ← 1 (integer);
        n ← 0 (integer);
        sum ← 0 (integer);
        Accept n;
        while(i <= n)
                sum ← sum + i;
                i ← i + 2;
        end while;
        Display sum; Stop
```

10. Write a pseudocode to find the reverse of a number. Store the reverse value in a different variable. Display the reverse.

```
Start
    digit ← 0 (integer);
    num ← 0 (integer);
    rev ← 0 (integer);

    Accept num;

    While(num > 0)
        digit ← num % 10;
        rev ← rev * 10 + digit;
        num ← num / 10;
    End while;

    Display rev;
Stop
```

11. Write a pseudocode to display a number in words.

Ex. 270176

Output: Two Seven Zero One Seven Six

```
Start
    digit ← 0 (integer);
    num ← 0 (integer);
    rev ← 0 (integer);
    Declare Array words of [10] (string) ←
        { "Zero", "One", "Two", "Three", "Four", "Five", "Six", "Seven", "Eight", "
        Nine" };
```

```
nonZeroFound ← false (boolean);
trailingZero, i ← 0 (integer);


Accept num;


While(num > 0)
    digit ← num % 10;
    if(nonZeroFound == false) then
        if (digit != 0)) then
            nonZeroFound ← true;
        else
            trailingZero ← trailingZero + 1;
         endif;
    endif;


    rev ← rev * 10 + digit;
    num ← num / 10;
End while;


While(rev > 0)
    Display words[(rev % 10) + 1];
    rev ← rev / 10;
End while;
```

```
        i ← 1;
        While(i <= trailingZero)
                Display "Zero";
                i ← i + 1;
        End while;
Stop
```

12. Write as many pseudocodes to generate the following series. In all the following cases, accept N:

4, 16, 36, 64, … N

```
Start
        i ← 2 (integer);
        n ← 0 (integer);


        Accept n;


        While( (i*i) <= n)
                Display i * i;
                i ← i + 2;
        End while;
Stop
```

1, -2, 3, -4, 5, -6, … N

```
Start
    i ← 1 (integer);
    n ← 0 (integer);

    Accept n;

    n ← abs(n);

    While(i <= n)
        if(i % 2 == 0)
            Display -i;
        else
            Display i;
        i ← i + 1;
    End while;
Stop

Start
    i ← 1 (integer);
    n ← 0 (integer);
    sign ← 1 (integer);

    Accept n;

    n ← abs(n);
```

```
        While(i <= n)
                Display i * sign;
                sign ← -sign;
                i ← i + 1;
        End while;
Stop
```

1, 4, 27, 256, 3125, … N

```
Start
        i ← 1 (integer);
        n ← 0 (integer);

        Accept n;

        While((i ^ i) <= n)
                Display (i ^ i);
                i ← i + 1;
        End while;
Stop
```

1, 4, 7, 12, 23, 42, 77, … N

```
Start
        i ← 1 (integer);
        j ← 4 (integer);
        k ← 7 (integer);
        next ← 0 (integer);
        n ← 0 (integer);

        Accept n;

        if(n >= 7) then
                Display i, j, k;
        else if(n >= 4) then
                Display i, j;
        else if(n >= 1) then
                Display i;
        next → i + j + k;
        While(next <= n)
                Display next;
                i ← j;
                j ← k;
                k ← next;

                next ← i + j + k;
        End while;
Stop
```

1, 4, 9, 25, 36, 49, 81, 100, … N

```
Start
        i ← 1 (integer);

        n ← 0 (integer);


        Accept n;


        While((i * i) <= n)

                if(i % 4 != 0) then

                        Display i * i;

                endif;

                i ← i + 1;

        End while;

Stop
```

```
Start
        i ← 1 (integer);

        n ← 0 (integer);


        Accept n;


        While((i * i) <= n)

                Display i * i;

                i ← i + 1;


                if(i % 4 == 0) then

                        i ← i + 1;
        endif;
    End while;
Stop
```

1, 5, 13, 29, 49, 77, … N

```
Start
    i ← 1 (integer);
    j ← 4 (integer);
    n ← 0 (integer);

    Accept n;

    While(i <= n)
        Display i;
        i ← i + j;
        j ← j + 4;

        if(j % 12 == 0) then
            j ← j + 4;
        endif;
    End while;
Stop
```

13. Write a pseudocode to find the sum of all the prime numbers in the range n to m. Display each prime number and also the final sum.

```
Start
    n ← 0 (integer);
    m ← 0 (integer);
    i ← 0 (integer);
    isPrime ← true (boolean);

    Accept n, m;

    if(n % 2 == 0) then
        n ← n + 1;
    endif;

    While(n <= m)
        isPrime ← true;
        i ← 3;
        while((i <= sqrt(n)) and (isPrime == true))
            if(n % i == 0) then
                isPrime = false;
            endif;
            i ← i + 1;
        end while;

        if(isPrime == true) then
            Display n;
```

```
                sum ← sum + n;
            endif;
            n ← n + 2;
        End while;
Stop
```

14. Write a pseudocode to find the factorial of a given number. 0! is always 1. Factorial of a negative number is not possible.

```
Start
        n ← 0 (integer);
        fact ← 1 (integer);

        Accept n;

        if (n < 0) then
            Display "Not possible";
        else
            While(n >= 2)
                fact ← fact * n;
                n ← n - 1;
            end while;
        endif;

        Display fact;
Stop
```

15. Write a pseudocode to accept a decimal number. Display it in the binary form.

```
Start
    n ← 0 (integer);

    bin ← 0 (integer);

    i ← 0 (integer);


    Accept n;


    While (n >= 1)

        bin → bin + (n % 2) * 10^i;

        n ← n / 2;

        i ← i + 1;

    end while;


    display bin;

Stop
```

16. Write a pseudocode to accept a binary number and display it in the decimal form.

```
Start
    dec ← 0 (integer);

    bin ← 0 (integer);

    i ← 0 (integer);


    Accept bin;
    While (n >= 1)
        dec → dec + (bin % 10) * 2^i;
        bin ← bin / 10;
        i ← i + 1;
    end while;

    display dec;
Stop
```

17. Write a pseudocode to display the 1st , 2nd , and 4th multiple of 7 which gives the remainder 1 when divided by 2,3,4,5 and 6

```
Start
    i ← 7 (integer);
    count ← 1 (integer);

    While (count <= 4)
        if(i % 2 == 1 and i % 3 == 1 and i % 4 == 1 and i % 5 == 1 and i % 6 == 1) then
                if(count !=3) then
                    display i;
                endif;
                count ← count + 1;
        endif;
        i ← i + 7;
    end while;
Stop
```

```
Start
    i ← 61 (integer);

    count ← 1 (integer);


    While (count <= 4)

        if(i % 7 == 0) then

                if(count !=3) then

                    display i;

                endif;

                count ← count + 1;

        endif;

        i ← i + 60;

    end while;

Stop
```

```
Start
    i ← 1 (integer);

    count ← 1 (integer);


    While (count <= 4)

        display (60 * (5 + (7 * (i-1)))) + 1;

        if(i == 3) then

            i ← i + 1;

        endif;

        i ← i + 1;

    end while;

Stop
```

```
Start
    i ← 0 (integer);

    count ← 1 (integer);


    While (count <= 4)

        display (301 + 420 * i);

        if(i == 3) then

            i ← i + 1;

        endif;


        i ← i + 1;

    end while;

Stop
```

18. Write a pseudocode to do the following:

Accept the item code, description, qty and price of an item. Compute the total for the item.

Accept the user's choice. If the choice is 'y' then accept the next set of inputs for a new item and compute the total. In this manner, compute the grand total for all the items purchased by the customer.

If the grand total is more than Rs. 10,000/- then, the customer is allowed a discount of 10%.

If the grand total is less than Rs. 1,000/- and the customer chooses to pay by card, then a surcharge of 2.5% is levied on the grand total.

Display the grand total for the customer.

```
Start
    itemCode, desc ← “ “ (string);
    qty ← 0 (integer);
    price ← 0.0 (double);
    granTot ← 0.0 (double);
    modePayment ← ‘R’ (char);
    choice ← ‘y’ (char);

    NOTE: ‘R’ indicates payment by card and ‘C’ indicates payment by cash.

    While(choice == ‘Y’ or choice == ‘y’)
        Accept itemCode, desc, qty, price;
        granTot ← granTot + (qty * price);

        Accept choice;
    End while;
    Accept paymentMode;
    if (granTot > 10000) then
        granTot ← granTot * 0.9);
    else
        If(granTot < 1000 and paymentMode == ‘R’)  then
             granTot ← granTot + (granTot * 0.025);
            endif;
        endif;
    endif;
    display granTot;
Stop
```

19. Write the pseudocodes to generate the following series. In all the following cases, accept N:

1, -2, 6, -15, 31, -56, … N

```
Start

    nTerms ← 0 (integer);

    i ← 0 (integer);

    a ← 1 (integer);

    sign ← 1 (integer);



    Accept nTerms;


    for i ← 1 to nTerms

        display a * sign;

        a ← a + i * i;

        sign ← -sign;

    end for;

Stop
```

1, 1, 2, 3, 5, 8, 13, … N

```
Start
    nTerms ← 0 (integer);
    i ← 0 (integer);
    a ← 1 (integer);
    b ← 0 (integer);
    next ← 1 (integer);

    Accept nTerms;

    for i ← 1 to nTerms
        Display next;
        a ← b;
        b ← next;
        next ← a + b;
    end for;
Stop
Start
    nTerms ← 0 (integer);
    i ← 0 (integer);
    a ← 1 (integer);
    b ← 1 (integer);

    Accept nTerms;

    Display a, b;
```

```
        for i ← 1 to nTerms
                b ← a + b;
                a ← b - a;

                display b;
        end for;
Stop

Start
        nTerms ← 0 (integer);
        i ← 0 (integer);
        a ← 1 (integer);
        b ← 1 (integer);

        Accept nTerms;

        Display a, b;

        for i ← 1 to nTerms / 2
                a ← a + b;
                b ← a + b;
                display a, b;
        end for;
Stop
```

1, -2, 4, -6, 7,-10, 10,-14… N

```
Start
    a ← 1 (integer);
    b ← -2 (integer);
    nTerms ← 0 (integer);
    i ← 0 (integer);

    Accept nTerms;

    for i ← 1 to nTerms
        display a,b;
        a ← a + 3;
        b ← b - 4;
    end for;
Stop
```

1, 5, 8, 14, 27, 49, … N

```
Start
    a ← 1 (integer);
    b ← 5 (integer);
    c ← 8 (integer);
    next ← 14 (integer);
    nTerms ← 0 (integer);
    i ← 0 (integer);

    Accept nTerms;
    for i ← 1 to nTerms
        display next;
        a ← b;
        b ← c;
        c ← next;
        next ← a + b + c;
    end for;
Stop
```

20. Write a pseudocode to find X^n (x to the power of n). Accept X and n. Display the result.

```
Start
    x ← 1 (integer);
    n ← 5 (integer);
    i ← 0 (integer);
    power ← 1 (integer);
    Accept x,n;
    If(x == 0 and n == 0) then
        Display “Not possible”;
    Else
        for i ← 1 to nTerms
            power ← power * x;
        end for;
    endif;
    Display power;
Stop
```

21. Write a pseudocode to display the reverse of a string.

```
Start
    str ← " " (string);
    rev ← " " (string);
    i ← 0 (integer);
    Accept str;
    for i ← 1 to length(str)
        rev[i] ← str[length(str) – i+1];
    end for;
Stop
Start
    str ← " " (string);
    rev ← " " (string);
    i ← 0 (integer);

    Accept str;

    for i ← 1 to length(str)
        rev ← str[i] + rev;
    end for;
Stop
```

22. Write a pseudocode to check if the string is a palindrome

```
Start
    str ← " " (string);
    rev ← " " (string);
    i ← 0 (integer);
    Accept str;
    for i ← 1 to length(str)
        rev ← str[i] + rev;
    end for;

    if(str == rev) then
        Display "Palindrome";
    else
        Display "Not a Palindrome";
    endif;
Stop
```

23. Write the pseudocodes to generate the following outputs. In all the following cases, accept N:

```
* * * * *
* * * * *
* * * * *
* * * * *
:
```

N rows

```
Start
    i, j ← 0 (integer);
    n ← 0 (integer);

    Accept n;

    for i ← 1 to n
        for j ← 1 to 5
            display " * ";
        end for;
        display;
    end for;
Stop
```

1 1 1 1 1

2 2 2 2 2

3 3 3 3 3

4 4 4 4 4

:

N rows

```
Start
    i, j ← 0 (integer);
    n ← 0 (integer);

    Accept n;
    for i ← 1 to n
        for j ← 1 to 5
            display i;
        end for;
        display;
    end for;
Stop
```

```
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
:
N rows
```

```
Start
    i, j ← 0 (integer);
    n ← 0 (integer);

    Accept n;

    for i ← 1 to n
        for j ← 1 to 5
            display j;
        end for;
        display;
    end for;
Stop
```

```
*
*   *
*   *   *
*   *   *   *
:
```

N rows

```
Start
    i, j ← 0 (integer);
    n ← 0 (integer);

    Accept n;

    for i ← 1 to n
        for j ← 1 to i
            display " * ";
        end for;
        display;
    end for;
Stop
```

24. Write the pseudocodes to generate the following outputs. In all the following cases, accept N:

```
1

1 2

1 2 3

1 2 3 4

:

N rows
```

```
Start
    i, j ← 0 (integer);
    n ← 0 (integer);

    Accept n;

    for i ← 1 to n
        for j ← 1 to i
            display j;
        end for;
        display;
    end for;
Stop
```

```
1

2 2
```

3 3 3

4 4 4 4

:

N rows

```
Start
        i, j ← 0 (integer);
        n ← 0 (integer);

        Accept n;

        for i ← 1 to n
                for j ← 1 to i
                        display i;
                end for;
                display;
        end for;
Stop
```

1

2 3

4 5 6

7 8 9 10

:

N rows

```
Start
    i, j ← 0 (integer);
    n ← 0 (integer);
    k ← 1 (integer);

    Accept n;

    for i ← 1 to n
        for j ← 1 to i
            display k;
            k ← k + 1;
        end for;
        display;
    end for;
Stop

Start
    i, j ← 0 (integer);
    n ← 0 (integer);

    Accept n;

    for i ← 1 to n
        for j ← 1 to i
            display (((i - 1) * i) / 2) + j;
        end for;
        display;
    end for;
Stop
```

1

1 2

3 5 8

:

:

N rows

```
Start
    i, j ← 0 (integer);
    n ← 0 (integer);
    a ← 1 (integer);
    b ← 0 (integer);
    next ← 1 (integer);

    Accept n;

    for i ← 1 to n
        for j ← 1 to i
            display next;
            a ← b;
            b ← next;
            next ← a + b;
        end for;
        display;
    end for;
Stop
```

25. Write the pseudocodes to generate the following outputs. In all the following cases, accept N:

```
1
-4   9
-16 25 -36
:
:
N rows
```

```
Start
    i, j ← 0 (integer);
    n ← 0 (integer);
    a ← 1 (integer);
    sign ← 1 (integer);

    Accept n;

    for i ← 1 to n
        for j ← 1 to i
            display a * a * sign;
            a ← a + 1;
            sign ← -sign;
        end for;
```

```
                display;
        end for;
Stop
```

```
1
1  2
6  24 120
:
:
N rows
```

```
Start
        i, j ← 0 (integer);
        n ← 0 (integer);
        a ← 1 (integer);
        b ← 0 (integer);
        Accept n;
        for i ← 1 to n
                for j ← 1 to i
                        display a;
                        b ← b + 1;
                        a ← a * b;
                end for;
                display;
        end for;
Stop
```

```
      *
    * *
  * * *
* * * *
:
```

N rows

```
Start
    i, j ← 0 (integer);
    n ← 0 (integer);

    Accept n;

    for i ← 1 to n
        for j ← 1 to n - i
            display “  “;
        end for;

        for j ← 1 to i
            display “ * “;
        end for;

        display;
    end for;
Stop
```

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

:

N rows

```
Start
    i, j ← 0 (integer);
    n ← 0 (integer);

    Accept n;

    for i ← 1 to n
        for j ← 1 to n - i
            display “  “;
        end for;

        for j ← 1 to 2 * i - 1
            display “ * “;
        end for;

        display;
    end for;
Stop
```

26. Write a pseudocode to store N elements in an array of integer. Display the elements. Accept a number to be searched. Display whether the number is found or not in the array (LINEAR SEARCH).

```
Start

    MAX_SIZE ← 1000 (integer);

    Declare Array a1 of [MAX_SIZE] of integer;

    item, n ← 0 (integer);

    found ← false (boolean);

    i ← 0 (integer);

    Accept n;


    for i ← 1 to n

        accept a1[i];

    end for;


    for i ← 1 to n

        display a1[i];

    end for;


    NOTE: LINEAR SEARCH algorithm


    Accept item;


    while((i <= n) and (found == false))

        if(item == a1[i]) then

            found ← true;

        endif;
```

```
            i ← i + 1;
        end while;

        if(found == true)
            Display "Item found";
        else
            Display "Item not found";
        endif;

Stop
Start
        MAX_SIZE ← 1000 (integer);
        Declare Array a1 of [MAX_SIZE] of integer;
        item, n ← 0 (integer);

        Accept n;

        for i ← 1 to n
            accept a1[i];
        end for;

        for i ← 1 to n
            display a1[i];
        end for;

        NOTE: LINEAR SEARCH algorithm
```

```
    Accept item;


    while((i <= n) and (a1[i] != item))
        i ← i + 1;
    end while;


    if(i <= n)
        Display "Item found";
    else
        Display "Item not found";
    endif;


Stop
```

27. Write a pseudocode to store N elements in an array of integer. Display the elements. Sort the elements. Accept a number to be searched. Display whether the number is found or not in the array using BINARY SEARCH.

```
Start
    MAX_SIZE ← 1000 (integer);
    Declare Array a1 of [MAX_SIZE] of integer;
    item, n ← 0 (integer);
    found ← false (boolean);
    low, high, mid ← 0 (integer);
    temp, i, j ← 0 (integer);


    Accept n;
```

```
for i ← 1 to n
        accept a1[i];
end for;

for i ← 1 to n
        display a1[i];
end for;
```

NOTE: Simple SELECTION SORT algorithm

```
for i ← 1 to n-1
        for j ← i+1 to n
                if(a1[i] > a1[j]) then
                        temp ← a1[i];
                        a1[i] ← a1[j];
                        a1[j] ← temp;
                endif;
        endfor;
end for;

Accept item;
```

NOTE: BINARY SEARCH algorithm

```
low ← 1;
```

```
    high ← n;
    mid ← (low + high) / 2;


    while((low <= high) and (found == false))
        if(item == a1[mid]) then
            found ← true;
        else if(item > a1[mid]) then
            low ← mid + 1;
        else
            high ← mid – 1;
        endif;
        endif;


        i ← i + 1;


        mid ← (low + high) / 2;
    end while;
    if(found == true)
        Display "Item found";
    else
        Display "Item not found";
    endif;


Stop
```

28. Write a pseudocode to store elements into a M * N matrix of integer. Display the matrix and its transpose.

```
Start
    MAX_SIZE ← 1000 (integer);
    Declare Array a1 of [MAX_SIZE,MAX_SIZE] of integer;
    m,n ← 0 (integer);
    i, j ← 0 (integer);

    Accept m, n;

    for i ← 1 to m
        for j ← 1 to n
            accept a1[i,j];
        end for;
    end for;

    for i ← 1 to m
        for j ← 1 to n
            display a1[i,j];
        end for;
        display;
    end for;

    for i ← 1 to n
        for j ← 1 to m
            display a1[j,i];
```

```
                end for;

                display;

        end for;

Stop
```

29. Write a pseudocode to store elements into a N * N matrix of integer. Display whether it is an identity matrix or not.

```
Start

        MAX_SIZE ← 1000 (integer);

        Declare Array a1 of [MAX_SIZE,MAX_SIZE] of integer;

        n ← 0 (integer);

        i, j ← 0 (integer);

        identity ← true (boolean);


        Accept n;


        for i ← 1 to n

                for j ← 1 to n

                        accept a1[i,j];

                end for;

        end for;


        for i ← 1 to n

                for j ← 1 to n

                        display a1[i,j];

                end for;

                display;
```

```
    end for;


    NOTE: Check for identity


    i ← 1;
    while((i<=n) and (identity == true))
        j ← 1;
        while((j<=n) and (identity == true))
            if(((i == j) and (a1[i,j] != 1)))  or
                (((i != j) and (a1[i,j] != 0))) then
                    identity ← false;
            endif;
            j ← j + 1;
        end while;
        i ← i + 1;
    end while;


    if(identity == true) then
        display "Identity matrix";
    else
        display "Not an Identity matrix";
    endif;


Stop
```

```
Start
    MAX_SIZE ← 1000 (integer);
    Declare Array a1 of [MAX_SIZE,MAX_SIZE] of integer;
    n ← 0 (integer);
    i, j ← 0 (integer);
    identity ← true (boolean);

    Accept n;

    for i ← 1 to n
        for j ← 1 to n
            accept a1[i,j];
        end for;
    end for;

    for i ← 1 to n
        for j ← 1 to n
            display a1[i,j];
        end for;
        display;
    end for;

    NOTE: Check for identity

    i ← 1;
    while((i<=n) and (identity == true))
```

```
          j ← 1;
          while((j<=i) and (identity == true))
                    if(((i == j) and (a1[i,j] != 1)))  or
                              (((i != j) and (a1[i,j] != 0)
                                        and (a1[j,i] != 0))) then
                                                  identity ← false;
                    endif;
                    j ← j + 1;
          end while;
          i ← i + 1;
     end while;

     if(identity == true) then
          display "Identity matrix";
     else
          display "Not an Identity matrix";
     endif;

Stop
```

30. Write a pseudocode to store elements into a N * N matrix of integer. Display whether it is a symmetric matrix or not.

```
Start
    MAX_SIZE ← 1000 (integer);
    Declare Array a1 of [MAX_SIZE,MAX_SIZE] of integer;
    n ← 0 (integer);
    i, j ← 0 (integer);
    symmtric ← true (boolean);

    Accept n;

    for i ← 1 to n
        for j ← 1 to n
            accept a1[i,j];
        end for;
    end for;

    for i ← 1 to n
        for j ← 1 to n
            display a1[i,j];
        end for;
        display;
    end for;

    NOTE: Check for Symmetry
```

```
        i ← 1;
        while((i<=n) and (symmtric == true))
            j ← 1;
            while((j<=i) and (symmtric == true))
                if(((i == j) and (a1[i,j] == 0)))  or
                        (((i != j) and (a1[i,j] != a1[j,i]))) then
                            symmtric ← false;
                endif;
                j ← j + 1;
            end while;
            i ← i + 1;
        end while;

        NOTE:
            symmetric = ((i==j) ?a1[i][j]) : a1[i][j] == a1[j][i]);

        if(symmtric == true) then
            display “Symmetric matrix”;
        else
            display “Not an Symmetric matrix”;
        endif;

    Stop
```