C PROGRAMMING LABORATORY

18CPL27

ATRIA INSTITUTE OF TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
Bengaluru - 560024
2020

C PROGRAMMING LABORATORY

Semester	**: I/II**	**CIE Marks**	**: 40**
Course Code	**: 18CPL17/27**	**SEE Marks**	**: 60**
Teaching Hours/week (L:T:P)	**: 0:0:2**	**Exam Hours**	**: 03**
	Credits : 01		

Course Learning Objectives:

This course **(18CPL17/27)** will enable students to:

- Write flowcharts, algorithms and programs.
- Familiarize the processes of debugging and execution.
- Implement basics of C programming language.
- Illustrate solutions to the laboratory programs.

Descriptions (if any):

- The laboratory should be preceded or followed by a tutorial to explain the approach or algorithm being implemented or implemented for the problems given.
- Note that experiment 1 is mandatory and written in the journal.
- Questions related with experiment 1, need to be asked during viva-voce for all experiments.
- Every experiment should have algorithm and flowchart be written before writing the program.
- Code should be traced using minimum two test cases which should be recorded.
- It is preferred to implement using Linux and GCC.

Laboratory Programs:

1. Familiarization with computer hardware and programming environment, concept of naming the program files, storing, compilation, execution and debugging, taking any simple C- code.

PART A

2. Develop a program to solve simple computational problems using arithmetic expressions and use of each operator leading to simulation of a commercial calculator. (No built-in math function)
3. Develop a program to compute the roots of a quadratic equation by accepting the coefficients. Print appropriate messages.
4. Develop a program to find the reverse of a positive integer and check for palindrome or not. Display appropriate messages.

5. An electricity board charges the following rates for the use of electricity: for the first 200 units 80 paise per unit: for the next 100 units 90 paise per unit: beyond 300 units Rs 1 per unit. All users are charged a minimum of Rs. 100 as meter charge. If the total amount is more than Rs 400, then an additional surcharge of 15% of total amount is charged. Write a program to read the name of the user, number of units consumed and print out the charges.
6. Introduce 1D Array manipulation and implement Binary search.
7. Implement using functions to check whether the given number is prime and display appropriate messages. (No built-in math function)

PART B

8. Develop a program to introduce 2D Array manipulation and implement Matrix multiplication and ensure the rules of multiplication are checked.
9. Develop a Program to compute Sin(x) using Taylor series approximation. Compare your result with the built- in Library function. Print both the results with appropriate messages.
10. Write functions to implement string operations such as compare, concatenate, string length. Convince the parameter passing techniques.
11. Develop a program to sort the given set of N numbers using Bubble sort.
12. Develop a program to find the square root of a given number N and execute for all possible inputs with appropriate messages. Note: Don't use library function sqrt(n).
13. Implement structures to read, write and compute average- marks and the students scoring above and below the average marks for a class of N students.
14. Develop a program using pointers to compute the sum, mean and standard deviation of all elements stored in an array of n real numbers.
15. Implement Recursive functions for Binary to Decimal Conversion.

Laboratory Outcomes:

The student should be able to:

- Write algorithms, flowcharts and program for simple problems.
- Correct syntax and logical errors to execute a program.
- Write iterative and wherever possible recursive programs.
- Demonstrate use of functions, arrays, strings, structures and pointers in problem solving.

Conduct of Practical Examination:

- All laboratory experiments, excluding the first, are to be included for practical examination.
- Experiment distribution
  - o For questions having only one part: Students are allowed to pick one experiment from the lot and are given equal opportunity.

- o For questions having part A and B: Students are allowed to pick one experiment from part A and one experiment from part B and are given equal opportunity.
- Strictly follow the instructions as printed on the cover page of answer script for breakup of marks
- Change of experiment is allowed only once and marks allotted for procedure part to be made zero.
- Marks Distribution (Subjected to change in accordance with university regulations)

a) For questions having only one part – Procedure + Execution + Viva-Voce: 15+70+15 = 100 Marks

b) For questions having part A and B
   i. Part A – Procedure + Execution + Viva = 4 + 21 + 5 = 30 Marks
   ii. Part B – Procedure + Execution + Viva = 10 + 49+ 11 = 70 Marks

Contents

1 Introduction

1.1 A1 Familiarization

Familiarization with computer hardware and programming environment, concept of naming the program files, storing, compilation, execution and debugging, taking any simple C-code.

1.2 Team

- Prof. Srinivasachar
- Prof. Satisha
- Prof. Kavya
- Prof. Deeksha

1.3 Lab Manual

- C Programming Laboratory
- 18CPL27
- Continuous Internal EValuation (CIE) 40 Marks
- Exam Part A - 40, Part B - 60
- 1, 2-7, 8-15 (14 experiments)

1.4 Lab Activities

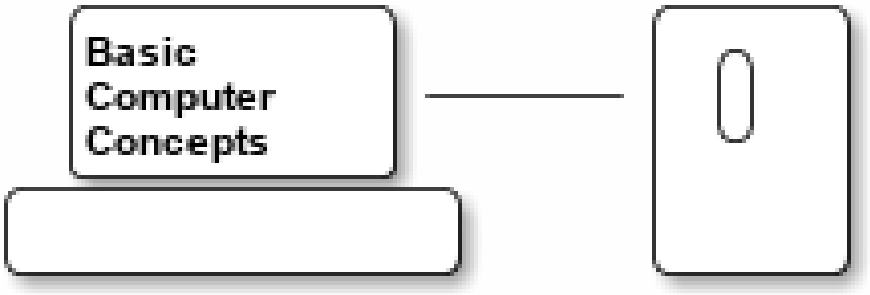
- Objectives
- Algorithm/Flow chart
- Program
- Edit
- Run
- Test against data different from the manual

1.5 Record

- Question
- Program
- Algorithm/Flowchart
- Procedure to run the program
- Test results

No.	Input	Output	Remarks
1			
2			
3			

1.6 Computer



1.7 Keyboard

- Alphabets Digits Delimitters Special symbols Function keys Keyboard matrix
- Tab Enter ^D ^C

1.8 Mouse

- Control the cursor
- Scroll the display
- Select the menu on left click
- Display properties on right click

1.9 Monitor

- Echo, program output (echo A)
- LCD/CRT
- Resolution
- Command terminal
- Touch screen
- xdpyinfo | grep 'dimensions:'
- GPU

1.10 Printer

- Impact printers dot matrix, daisy wheel
- Non impact printers inkjet, laser (image on drum transferred to paper)

1.11 Unix commands

1. mkdir -p D/<name>

2. cd ~/D/name
3. pwd
4. ls -l
5. ^D is EOF on unix unlike ^Z on windows
6. ^c to terminate the running program

1.12 Debugging

1. print statements
2. tracing

gdb commands

- gdb a.out
- b main | line
- run
- n next
- p variable

1.13 Continuous evaluation

1. Write the program of the previous experiment
2. Analyse the question
3. Design
4. Develop
5. Test
6. Viva

1.14 Persistence

1. Memory (volatile)
2. Disk file (persistant is echo.c)

PART A

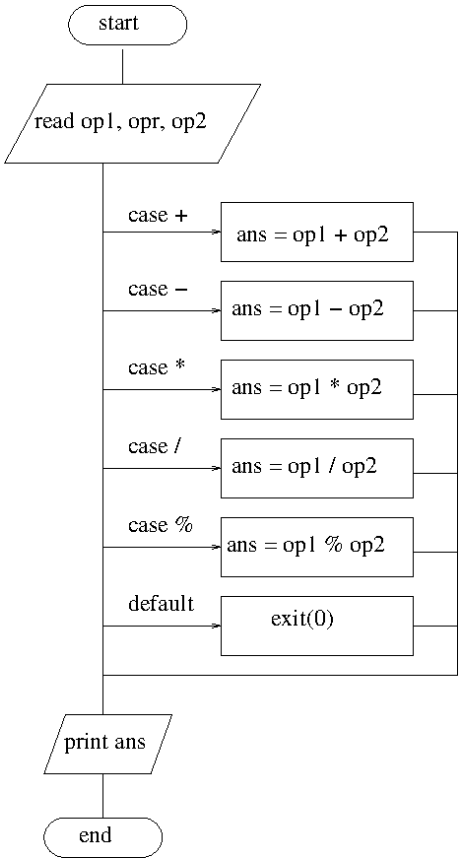
2 A2 Calculator

Develop a program to solve simple computational problems using arithmetic expressions and use of each operator leading to simulation of a commercial calculator. (No built-in math function)

2.1 Algorithm

1. Prompt for expression e.g op1+op2 w/o spaces
2. Read op1, opr, op2
3. Compute answer based on 'opr'
4. '+': answer = op1 + op2
5. '-': answer = op1 - op2
6. '*': answer = op1 * op2
7. '/': answer = op1 / op2
8. '%': answer = op1 % op2
9. default
   printf("invalid operator")
   exit(0)
10. printf(answer)

2.2 Flowchart



2.3 C Program

```c
/*2. Calculator, filename calc.c  */
#include<stdio.h>

int main()
{
  int op1, op2, ans;
  char opr;

  printf("Enter expression[e.g. 2+3]:");
  scanf("%d%c%d", &op1, &opr, &op2);

  switch(opr)
    {
    case '+':
      ans = op1+op2;
      break;

    case '-':
      ans = op1 - op2;
      break;

    case '*':
      ans = op1*op2;
      break;

    case '/':
      ans = op1/op2;
      break;

    case '%':
      ans = op1%op2;
      break;
    }
  printf("Answer = %d\n", ans);
}
```

2.4 Execution steps

1. gedit calc.c
2. gcc -g calc.c
3. ./a.out

2.5 Test Results

No	Expr	Value
1	10+5	15
2	10-5	5
3	10*5	50
4	10/5	2
5	10%5	0

2.6 Exceptions

1. Integer division
2. No default case

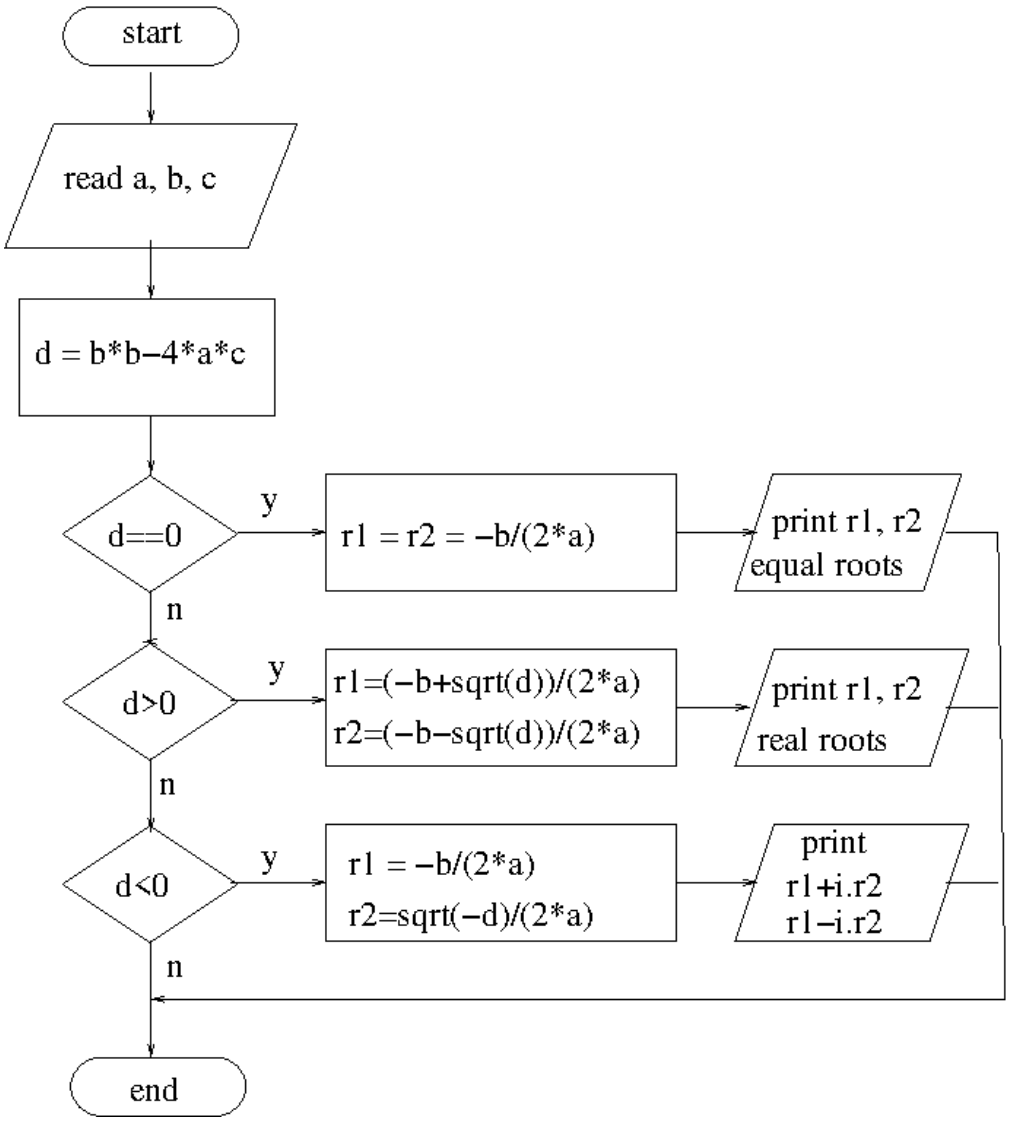
3 A3 Qudratic equation

Develop a program to compute the roots of a quadratic equation by accepting the coefficients. Print appropriate messages.

3.1 Algorithm

1. Prompt for the coeeficients of quadratic equation.
2. Read the coeeficients a, b and c
3. Find the discriminant and consider cased d = 0, d > 0 and d < 0
4. d = 0 roots are equal and they are -b/2a
5. d > 0, roots are real and they are (-b $± $ sqrt(d))/2a
6. d < 0, roots are imaginary and tey are (-b $± $ +i.fabs(sqrt(d)))/2a

3.2 Flowchart



3.3 C Program

```c
/*3. Quadratic equation, filename quad.c */
#include <stdio.h>
#include <math.h>

int main()
{
  float a, b, c, d, r1, r2;

  printf("Enter the coefficients a, b and c: ");
  scanf("%f%f%f", &a, &b, &c);

  d = b*b - 4*a*c;

  if (d == 0)
    {
      r1 = r2 = -b/(2*a);
      printf("Roots are equal: r1 = %f, r2=%f\n", r1, r2);
    }
  else if (d > 0)
    {
      r1 = (-b + sqrt(d))/(2*a);
      r2 = (-b - sqrt(d))/(2*a);
      printf("Roots are real: r1 = %f, r2=%f\n", r1, r2);
    }
  else if (d < 0)
    {
      r1 = -b/(2*a);
      r2 = sqrt(fabs(d))/(2*a);
      printf("Roots are complex: r1 = %f + i%f, r2=%f - i%f\n", r1, r2, r1, r2);
    }
}
```

3.4 Execution steps

1. gedit calc.c
2. gcc -g calc.c -lm
3. ./a.out

3.5 Test Results

No	Input a, b, c	Output r1, r2	Remark
1	1, 2, 1	r1 = +1, r2 -1	b^2 - 4ac = 0
2	1, 3, 1	r1 = -1, r2 = -2	b^2 - 4ac > 0
3	1, 2, 2	r1 = -1 + i1, -1 - i.1	b^2 - 4ac < 0

3.6 Exceptions

1. Cannot have roots when a = 0. But ignoring it.
2. Danger in comparing float d with inter 0. But ignoring it.

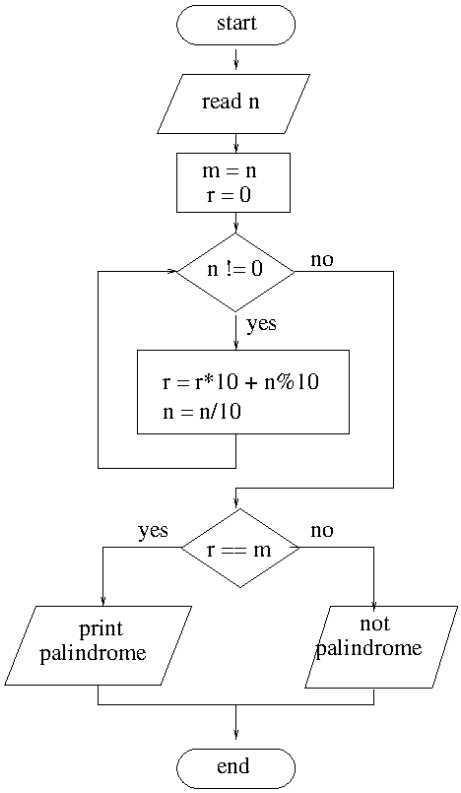
4 A4 Palindrome

Develop a program to find the reverse of a positive integer and check for palindrome or not. Display appropriate messages.

4.1 Algorithm

1. Promt for the number
2. Read the number
3. Copy the number to variable say 'n' for modifications
4. Compute the least significant digit
5. Compute the rev number using horner's rule r = r + d*10
6. Divide the number n by 10
7. Repeat steps 4, 5, 6 til 'n' not zero

4.2 Flowchart



4.3 C Program

```c
/*4. Palindrome */
#include<stdio.h>

int main()
{
  int n, m, r;

  printf(" Enter a number:");
  scanf("%d", &n);

  m = n;
  r = 0;

  while(n != 0)
    {
      r = r*10 + n%10;
      n = n/10;
    }

  if (r == m)
    printf("palindrome\n");
  else
    printf("not a palindrome");

}
```

4.4 Execution steps

1. gedit calc.c
2. gcc -g calc.c -lm
3. ./a.out

4.5 Test Results

Test the following expressions

No	Input	Output	Remarks
1	121	palindrome	
2	122	not a palindrom	

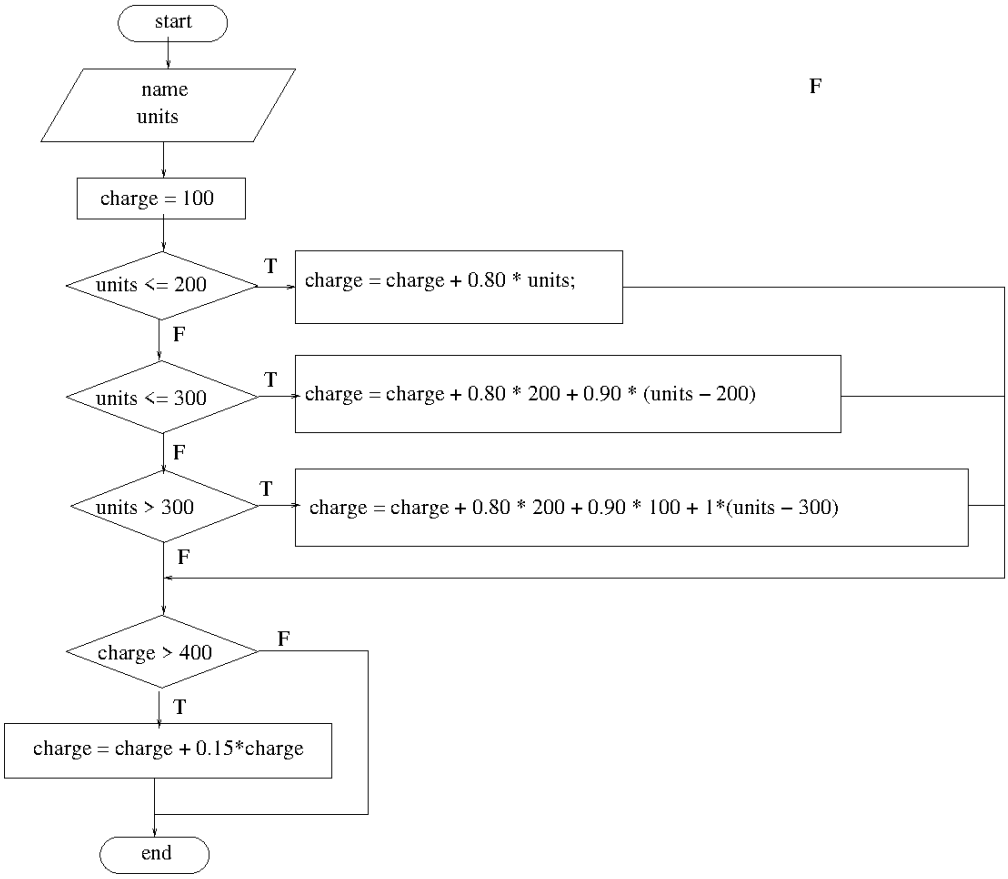
5 A5 Electricity Bill

An electricity board charges the following rates for the use of electricity: for the first 200 units 80 paise per unit: for the next 100 units 90 paise per unit: beyond 300 units Rs 1 per unit. All users are charged a minimum of Rs. 100 as meter charge. If the total amount is more than Rs 400, then an additional surcharge of 15% of total amount is charged. Write a program to read the name of the user, number of units consumed and print out the charges.

5.1 Algorithm

1. Prompt for name and units consumed
2. Read name and units
3. Minimum charge = 100
4. Upto 200 units, minimum charge + (units - 200)*0.80
5. Upto 300 units, minimum charge + 200*0.8 + (unit-300)*0.9
6. Beyond 300 units, minimum charge + 200*0.8 + 100*0.9 + (unit-300)*1
7. Bill is charge + 15% surcharge
8. Print the name and the charge

5.2 Flowchart



5.3 C Program

```c
/*5. Electricity bill */
#include<stdio.h>

int main()
{
  char name[30];
  int units;
  float charge;

  printf("Enter Consumer Name and units consumed:");
  scanf("%s%d", name, &units);

 charge = 100;

 if (units <= 200)
   {
     charge += 0.80*units;
   }
 else if (units <= 300)
   {
     charge += 0.80*200 + 0.90*(units-200);
   }
 else
   {
     charge +=0.80*200 + 0.90*100 + 1*(units-300);
   }

 if (charge > 400)
  {
    charge += 0.15*charge;
  }

 printf("%.2f\n", charge);
}
```

5.4 Execution steps

1. Start the terminal
2. Start the editor gedit
3. Edit the program 5.c
4. Compile on the terminal gcc -g 5.c
5. Run the program ./a.out

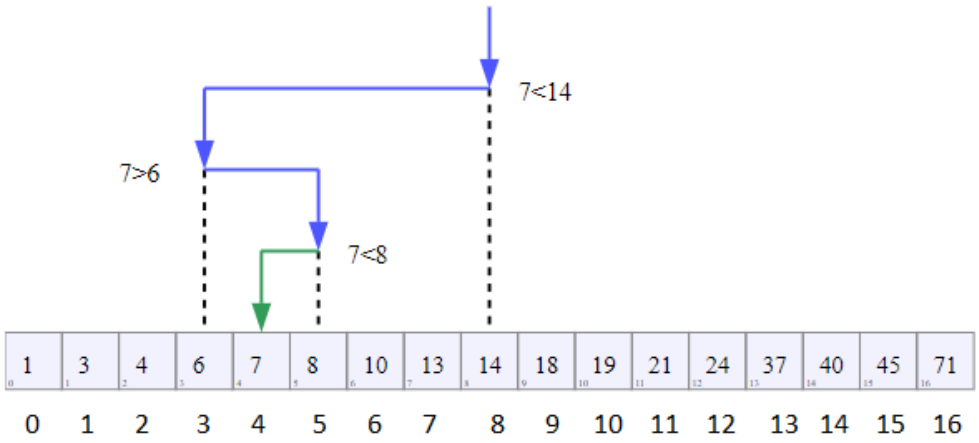
5.5 Test results

No	Input	Output	Remarks
1	X 150	220.0	
2	Y 250	305.0	
3	Z 350	400.0	
4	A 450	575.00	

6 A6 Binary search

Introduce 1D Array manipulation and implement Binary search.

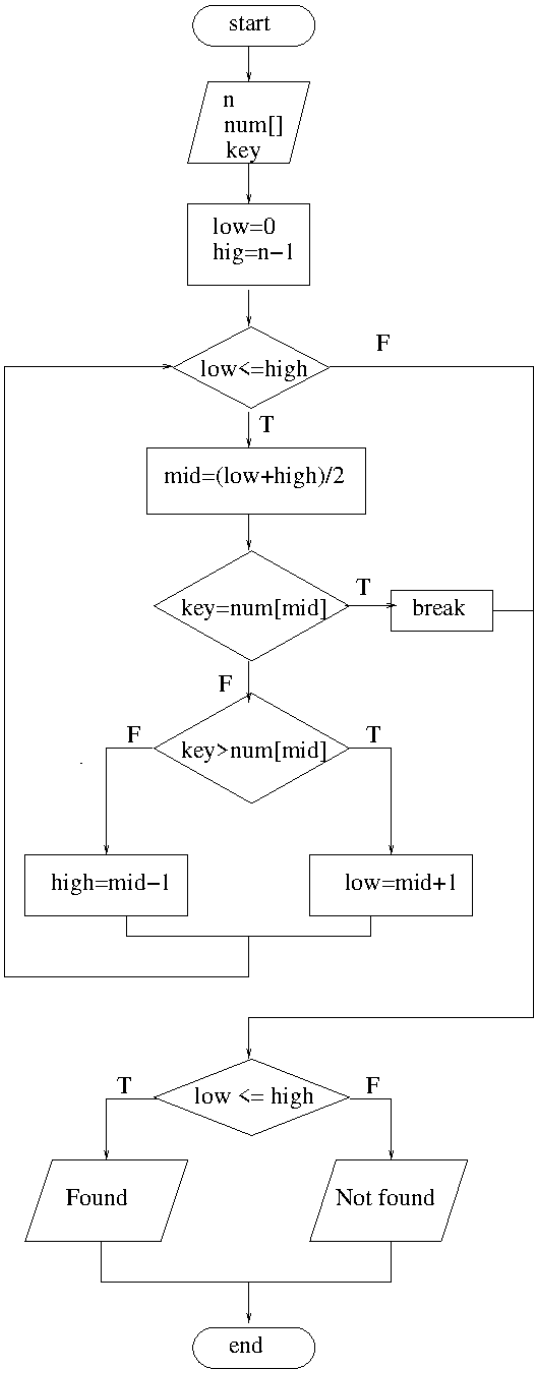
6.1 Background



6.2 Algorithm

1. Prompt for 'n', size of array
2. read the number 'n'
3. Prompt for array of numbers num[]
4. Read the numbers num[]
5. Prompt for the 'key'
6. Read the key
7. Note down the array limits low=0, high = n-1
8. While low is les than or equal to high do steps 9-12
9. Compute index mid = (low + high)/2
10. if key = a[mid] then key is found at the position mid.
11. Compare key and mid and search in the first or second half.
12. if key < a[mid] the high = mid-1 else low = mid+1
13. Print the key and found at mid

6.3 Flowchart



6.4 C Program

```c
/*6. Binary Search */
#include<stdio.h>
void main()
{
  int num[10];
  int n, key, low, high, mid;

  printf("Enter n, size of array:");
  scanf("%d", &n);

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

  printf("Enter the key:");
  scanf("%d", &key);

  low =0;
  high = n-1;

  while(low <= high)
    {
      mid=(low+high)/2;
      if (key == num[mid]) break;

      if (key > num[mid])
        low = mid+1;
      else
        high = mid-1;
    }

  if (low <= high)
    printf("Key %d found at position %d", key, mid+1);
  else
    printf("key %d not found", key);
}
```

6.5 Execution steps

1. Start the terminal
2. Start the editor gedit
3. Edit the program 6.c
4. Compile on the terminal gcc -g 6.c
5. Run the program ./a.out

6.6 Test results

No	Input	Output	Remarks
1	5 1 2 3 4 5 3	Key 3 found at position 3	
2	5 1 2 3 4 5 7	Key 7 Not found.	

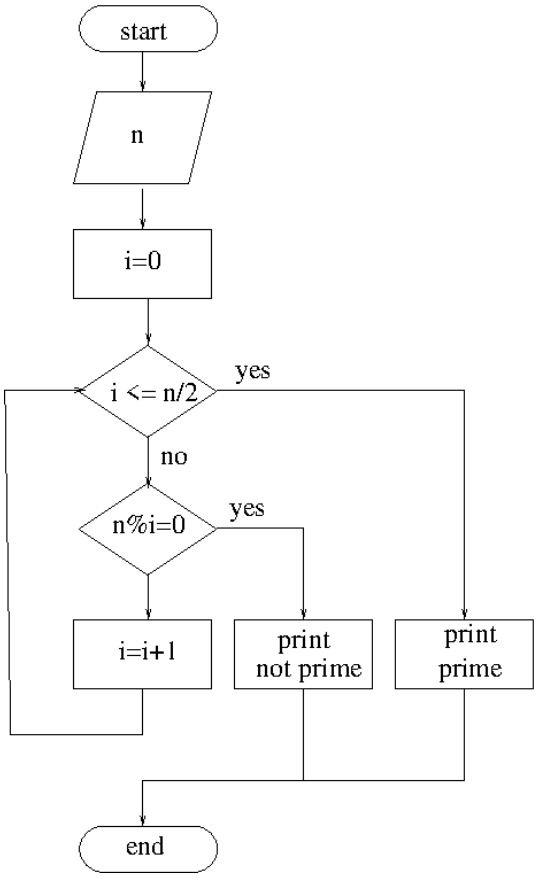
7 A7 Prime

Implement using functions to check whether the given number is prime and display appropriate messages. (No built-in math function)

7.1 Algorithm

1. Prompt for a number 'n'
2. Read the number n
3. divide the number 'n' by the numbers 2 to n/2
4. Any of the reminders is zero means the number is NOT a prime
5. else the number is prime

7.2 Flowchart



7.3 C Program

```c
/*7. Prime */
#include<stdio.h>

int isprime(int n)
{
  for(int i=2; i<= n/2; i++)
    {
      if (n % i == 0) return 0;
    }
  return 1;
}

int main()
{
  int n;

  printf("Enter a number:");
  scanf("%d", &n);

  if (isprime(n) == 1)
    printf("%d is a Prime number", n);
  else
    printf("%d is not a prime number", n);
}
```

7.4 Execution steps

1. Start the terminal
2. Start the editor gedit
3. Edit the program 7.c
4. Compile on the terminal gcc -g 7.c
5. Run the program ./a.out

7.5 Test results

No	Input	Output	Remarks
1	2	2 is prime	
2	4	4 is not prime	
3	7	7 is prime	

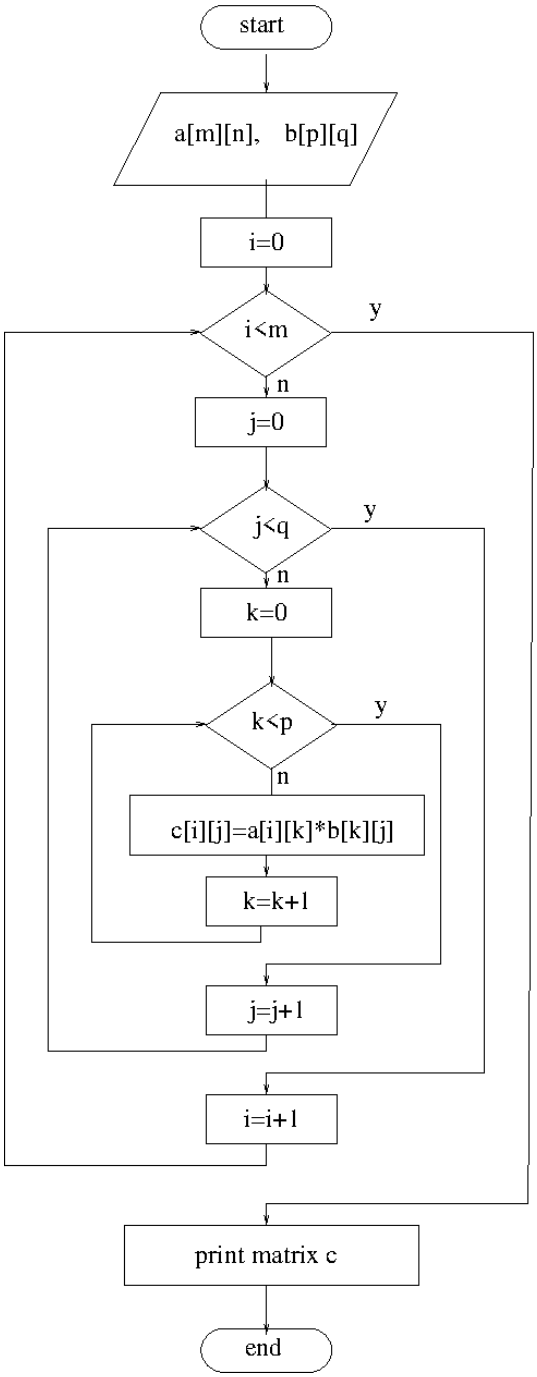
8 B1 Matrix multiplication

Develop a program to introduce 2D Array manipulation and implement Matrix multiplication and ensure the rules of multiplication are checked.

8.1 Algorithm

1. Prompt for the order of the matrix A[m, n]
2. Read m and n
3. Prompt for the order of the matrix B[p, q]
4. Read p and q
5. Columns of A and rows of B must be same. Otherwise error.
6. Compute C[i,j] is multiplication of i-th row of A and j-th column of B
7. Print the resultant matrix C

8.2 Flowchart



8.3 C Program

```c
/* Matrix multiplication */
#include <stdio.h>

int main()
{
  int a[10][10], b[10][10], c[10][10];
  int m, n;
  int p, q;

  printf("Enter the order of matrix A - m, n:");
  scanf("%d%d", &m, &n);

  printf("Enter the order of matrix B - p, q:");
  scanf("%d%d", &p, &q);

  if (n != p)
    {
      printf("Multipliction is not possible:");
      return 1;
    }

  printf("Enter matrix A[m ,n]:\n");
  for(int i=0; i < m; i++)
    for(int j=0; j < n; j++)
      scanf("%d", &a[i][j]);

  printf("Enter matrix B[p, q]:\n");
  for(int i=0; i < p; i++)
    for(int j=0; j < q; j++)
          scanf("%d", &b[i][j]);

  //c[m,q] = a[m,n] X b[p,q]; n = p
  for(int i=0; i < m; i++) //rows of a(m)
    {
      for(int j=0; j < q; j++) //cols of b(q)
        {
          c[i][j]=0;
          for(int k=0; k < p; k++) //cols of a(n) = rows of b(p)
            c[i][j] += a[i][k]*b[k][j]; // row a X col b
        }
    }

  printf("Resultant Matrix C:\n");
  for(int i=0; i < m; i++)
    {
      for(int j=0; j < q; j++)
```

```
        printf("%d\t", c[i][j]);
      printf("\n");
    }
}
```

8.4 Execution steps

1. Start the terminal
2. Start the editor gedit
3. Edit the program 8.c
4. Compile on the terminal gcc -g 8.c
5. Run the program ./a.out

8.5 Test results

Input		Output
a	b	c
mxn	pxq	mxq
3x2	2x4	3x4
1 3	1 3 2 2	7 15 17 5
2 4	2 4 5 1	10 22 24 8
2 5		12 26 29 9

9 B2 Sin(x)

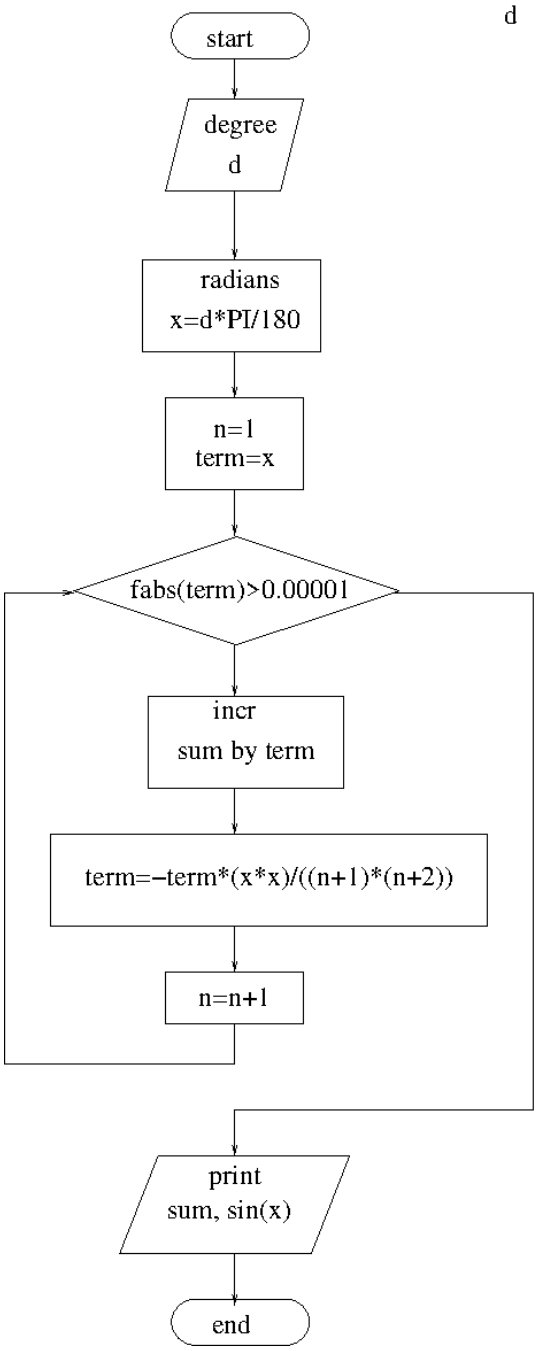
Develop a Program to compute Sin(x) using Taylor series approximation. Compare your result with the built-in Library function. Print both the results with appropriate messages.

9.1 Algorithm

1. Prompt for angle in degrees
2. Read the degree
3. Convert degree into x radians. 1 degree = $\pi/180$
4. $$sin(x) = \frac{1}{1!}x^1 + \frac{-1}{3!}x^3 + \frac{1}{5!}x^5 + ...$$
5. First term is x/1
6. Next term
$$T_n = (-1).T_{n-1} \times \frac{x^2}{(n+1)(n+2)}$$
7. Add the term to sum until the term value is negligible
8. Print sum as value of sin(x). Also sin(x) using the library funtion

9.2 Flowchart

d



9.3 C Program

```c
/*9. sin(x) */
#include <stdio.h>
#include <math.h>

int main()
{
  int n, degree;
  float x, sum=0, term;

  printf("Enter angle in degrees:");
  scanf("%d", &degree);

  x = degree*(3.142/180);
  n = 1;

  /* sin(x) = $\frac{1}{1!}x^1 + \frac{-1}{3!}x^3 + \frac{1}{5!}x^5 + ... */

  term = x/n;
  while(fabs(term) >= 0.00001)
  {
    sum = sum+term;
    term = -term*(x*x)/((n+1)*(n+2));

    n++;
  }

  printf("%.2f, %.2f\n", sum, sin(x));
}
```

9.4 Execution steps

1. Start the terminal
2. Start the editor gedit
3. Edit the program 9.c
4. Compile on the terminal gcc -g 9.c
5. Run the program ./a.out

9.5 Test results

No	Input	Output	Remarks
1	90	1.04, 1.00	
2	0	0.00, 0.00	
3	60	0.87, 0.87	

10 B3 Strings

Write functions to implement string operations such as compare, concatenate, string length. Convince the parameter passing techniques.

10.1 Algorithm

1. prompt for two strings
2. Read the strings

length(string)

1. Count the chars in the string until null char
2. return the count as length

concatenate(string 1, string 2)

1. Append string2 to string1

compare(string 1, string 2)

1. Strings are different if the string lengths are not same
2. Strings are same if the corresponding chars are same in both the strings
3. Return 1 if any char is different. Otherwise 0

10.2 C Program

```c
/*10. String methods */
#include <stdio.h>

int mystrlen(char s[])
{
  int i;

  for(i=0; s[i]; i++);
  return i;
}
```

```c
void mystrcat(char s1[], char s2[])
{
  int m, n;

  m = mystrlen(s1);
  n = mystrlen(s2);

  for(int i=0; i <= n; i++) s1[m++] = s2[i];
}

int mystrcmp(char s1[], char s2[])
{
  int m, n;

  m = mystrlen(s1);
  n = mystrlen(s2);

  if(m != n) return 1;

  for(int i=0; i <= n; i++)
    if(s1[i] != s2[i]) return 1;

  return 0;
}

int main()
{
  char s1[100], s2[100];
  int m, n;

  puts("Enter s1:");
  gets(s1);

  puts("Enter s2:");
  gets(s2);

  m = mystrlen(s1);
  n = mystrlen(s2);

  printf("strlen:%d,%d\n", m, n);

  if(mystrcmp(s1, s2) == 0) printf("strcmp: same\n");
  else printf("strcmp:different\n");

  mystrcat(s1, s2);
  printf("strcat:%s\n", s1);
}
```

10.3 Execution steps

1. Start the terminal
2. Start the editor gedit
3. Edit the program 10.c
4. Compile on the terminal gcc -g 10.c
5. Run the program ./a.out

10.4 Test results

No	Input	Output	Remarks
1	abc 123	strlen:3,3 strcmp:different strcat:abc123	
2	abc abc	strlen:3,3 strcmp:same strcat:abcabc	

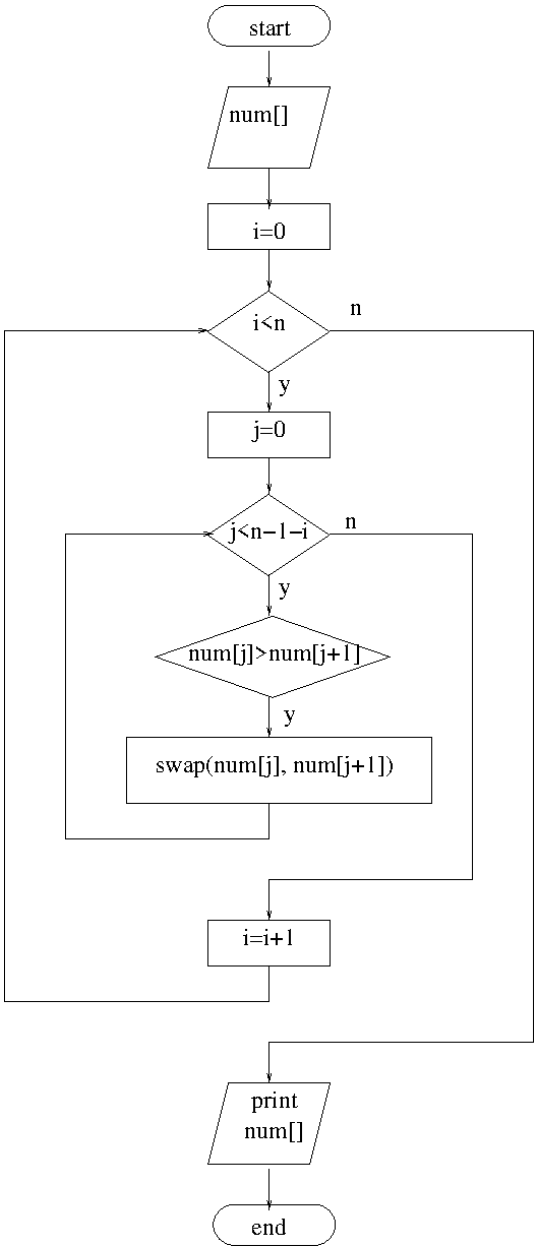
11 B4 Buble sort

Develop a program to sort the given set of N numbers using Bubble sort.

11.1 Algorithm

1. Prompt for the number 'n'
2. Read the numbers a[]
3. Swap the adjacent numbers of a[] if not in order
4. Each iteration (i.e step 3) pushes the largest number to the end of a[]
5. Reduce the size of array a[] by i iterations
6. After n iterations the numbers have been sorted

11.2 Flowchart



11.3 C Program

```c
/* Bubble sort */
#include<stdio.h>

int main()
{
  int num[25];
  int n, temp;

  printf("Enter n:");
  scanf("%d", &n);

  printf("Enter the array:");
  for(int i=0; i < n; i++)
    {
      scanf("%d", &num[i]);
    }

  for(int i=0; i < n; i++)
    {
      for(int j=0; j < n-1-i; j++) /* i-th iteration means i numbers sorted */
        {
          if(num[j] > num[j+1])
            {
              temp = num[j];
              num[j] = num[j+1];
              num[j+1] = temp;
            }
        }
    }

  printf("Sorted numbers are:");
  for(int i=0; i < n; i++)
    {
      printf("%d ", num[i]);
    }

  printf("\n");
}
```

11.4 Execution steps

1. Start the terminal
2. Start the editor gedit
3. Edit the program 11.c
4. Compile on the terminal gcc -g 11.c
5. Run the program ./a.out

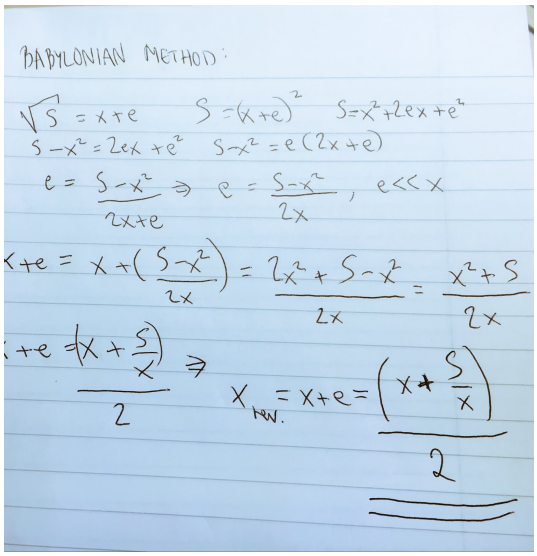
11.5 Test results

No	Input	Output	Remarks
1	1 5 3 2 4	1 2 3 4 5	
2	1 2 3 4 5	1 2 3 4 5	
2	5 4 3 2 1	1 2 3 4 5	

12 B5 Square root

Develop a program to find the square root of a given number and execute for all possible inputs with appropriate messages. Note: Don't use library function sqrt(n).

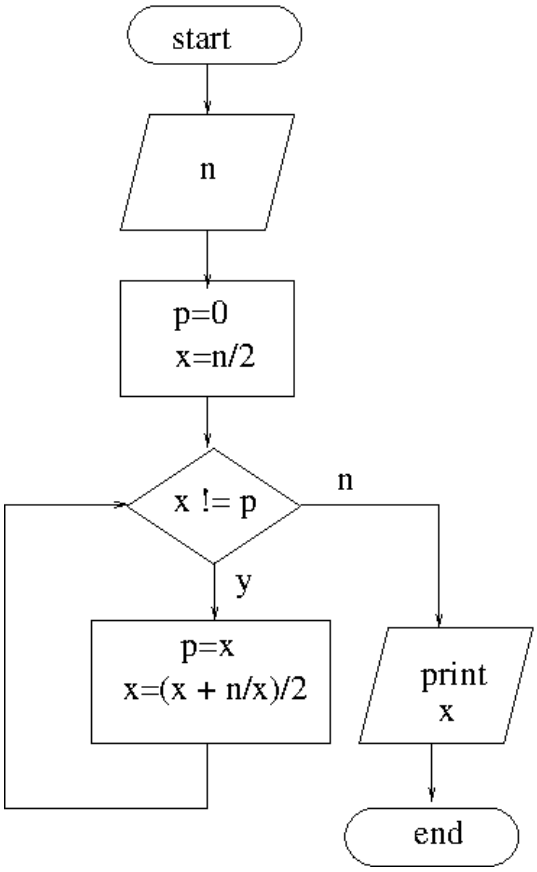
12.1 Divide and average method



12.2 Algorithm

1. Guess the root x as n/2
2. Compute the successive root x = (x + n/x)/2
3. Stop if there is no change in x
4. Print x as the root

12.3 Flowchart



12.4 C Program

```c
/*12. Square root */
#include <stdio.h>

int main()
{
  float n, x, p;

  printf("Enter n:");
  scanf("%f", &n);

  p = 0;
  x = n/2;

  while(x != p)
    {
      p = x;
      x = (x + n/x)/2;
    }

  printf("%.0f", x);
}
```

12.5 Execution steps

1. Start the terminal
2. Start the editor gedit
3. Edit the program 12.c
4. Compile on the terminal gcc -g 12.c
5. Run the program ./a.out

12.6 Test results

No	Input	Output	Remarks
1	16	4	
2	25	5	
3	256	16	

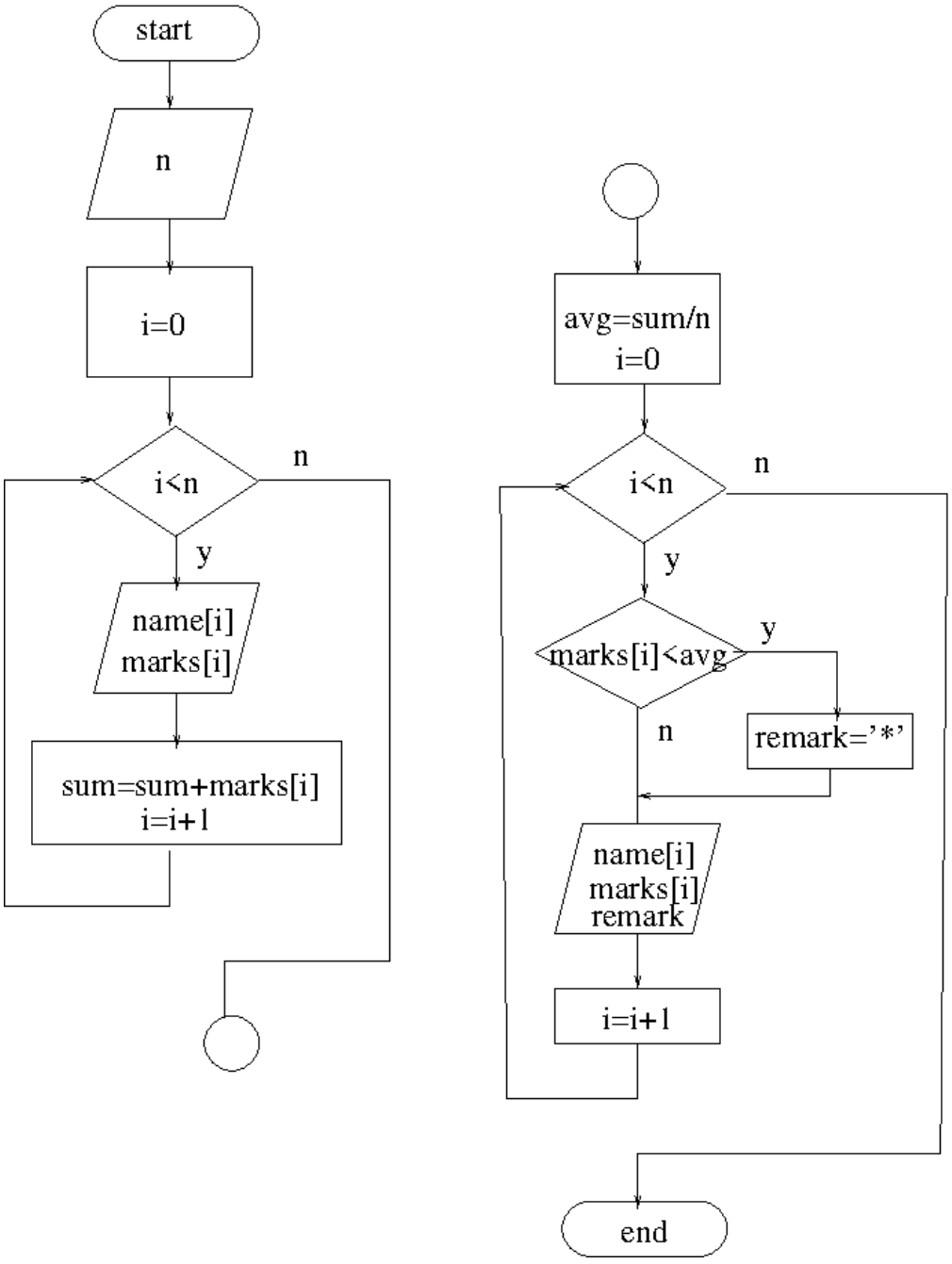
13 B6 Statistics

Implement structures to read, write and compute average-marks and the students scoring above and below the average marks for a class of N students.

13.1 Algorithm

1. Define student structure with name and marks
2. Allocate memory for 10 students
3. Prompt for number of students
4. Read student details name and marks for eacxh student
5. Compute sum of marks of all students
6. Compute average marks of n students
7. Print student name, marks and remark '<' if mark < average else is '>'.

13.2 Flowchart



13.3 C Program

```c
/* Statistics */
#include<stdio.h>

struct STUDENT
{
  char name[20];
  int marks;
};

struct STUDENT s[10];

int main()
{
  int n, sum, l, h;
  float avg;

  printf("Enter the number of students:");
  scanf("%d", &n);

  printf("Enter student name and marks:\n");
  for(int i=0; i<n; i++)
    {
      scanf("%s%d", s[i].name, &s[i].marks);
    }

  printf("Students data:\n");
  for(int i=0; i<n; i++)
    {
      printf("%s %d\n", s[i].name, s[i].marks);
    }

  sum = 0;
  for(int i=0; i<n; i++)
    {
      sum = sum + s[i].marks;
    }

  avg = (float)sum/n;

  l = h = 0;
  for(int i=0; i<n; i++)
    {
      s[i].marks < avg ? l++ : h++;
    }
  printf("sum = %d, avg = %.2f, lower = %d, higher = %d\n", sum, avg, l, h);
}
```

13.4 Execution steps

1. Start the terminal
2. Start the editor gedit
3. Edit the program 13.c
4. Compile on the terminal gcc -g 13.c
5. Run the program ./a.out

13.5 Test results

No	Input	Output	Remarks
1	3	s1 5	
	s1 5	s2 6	
	s2 6	s3 7	
	s3 7	sum = 18, avg = 6.00, lower = 1, higher =2	

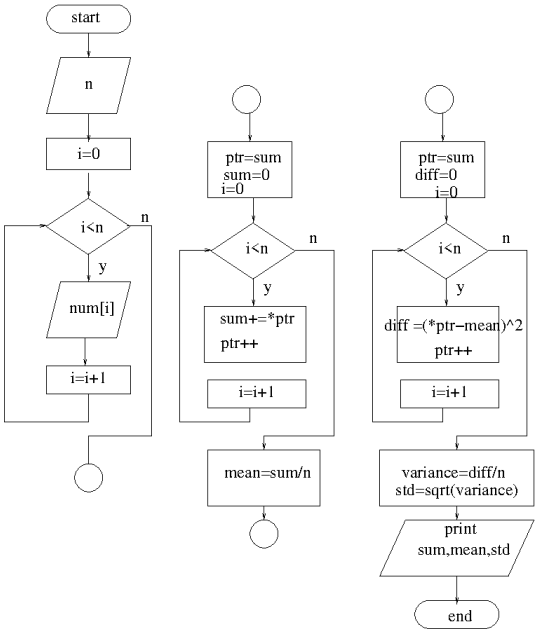
14 B7 Pointers

Develop a program using pointers to compute the sum, mean and standard deviation of all elements stored in an array of n real numbers.

14.1 Algorithm

1. Prompt for number of elements
2. Read the number 'n'
3. Read the array of numbers
4. Compute sum of the numbers in the array
5. Compute mean = sum/n
6. Compute sum of squared differences i.e (n - mean)^2
7. Compute the variance = sum/n
8. Compute stdev = sqrt(variance)

14.2 Flowchart



14.3 C Program

```c
/*14. Pointers */
#include<stdio.h>
#include<math.h>

int main()
{
  int n;
  float a[10], *p;
  float sum = 0, dev = 0, mean, std;

  printf("Enter n:");
  scanf("%d", &n);

  printf("Enter a:");
  for(int i=0; i<n; i++)
    {
      scanf("%f", &a[i]);
    }

  p = a;
  for(int i=0; i<n; i++)
    {
      sum += *p;
      p++;
    }
  mean = sum/n;

  p = a;
  for(int i=0; i<n; i++)
    {
      dev += (*p - mean)*(*p - mean);
      p++;
    }
  std = sqrt(dev/n);

  printf("Sum = %f, Mean = %f, std = %.3f\n", sum, mean, std);
}
```

14.4 Execution steps

1. Start the terminal
2. Start the editor gedit
3. Edit the program 14.c
4. Compile on the terminal gcc -g 14.c
5. Run the program ./a.out

14.5 Test results

No	Input	Output	Remarks
1	5		
	1 2 3 4 5	Sum = 15.0, Mean = 3.0, Std = 1.414	
2	5		
	1 1 1 1 1	Sum = 5.0, Mean = 1.0, Std = 0.0	

15 B8 Recursion

Implement Recursive functions for binary to decimal conversion.

15.1 Background

```
101
2*(10) + 1
2*(2*(1) + 0) + 1
5

2*(n/10) + n%10
```

15.2 Algorithm

recursion is d(n) = d(n/2)*2 + n%2

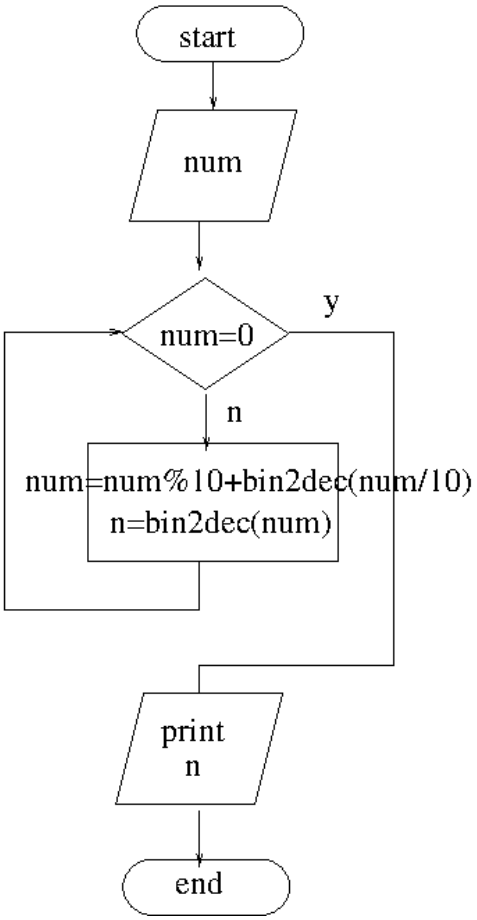
bd(n)

1. Base step: if n is 0 then return 0.
2. Recursion step: return 2*bd(n/10) + n%10

main()

1. Prompt for binary number
2. Read the number
3. Invoke bd(n)
4. Print the result

15.3 Flowchart



15.4 C Program

```c
/*15. Recursion */

#include <stdio.h>

int bd(int n)
{
  if (n == 0) return (0);
  return 2*bd(n/10) + n%10;
}

int main()
{
  int n;

  printf("Enter n:");
  scanf("%d", &n);

  printf("%d\n", bd(n));
}
```

15.5 Execution steps

1. Start the terminal
2. Start the editor gedit
3. Edit the program 15.c
4. Compile on the terminal gcc -g 15.c
5. Run the program ./a.out

15.6 Test results

No	Input	Output	Remarks
1	101	5	
2	1111	15	