STUDENT LAB MANUAL

PROGRAMMING FOR PROBLEM SOLVING LAB
18MCA115

R-18

J SHEIK MOHAMED
Asst. Professor / MCA

MCA DEPARTMENT
SREENIVASA INSTITUTE OF TECHNOLOGY AND MANAGEMENT STUDIES
(AUTONOMOUS)
(Approved by AICTE, New Delhi, Affiliated to JNTUA, Anantapuramu, Accredited by NAAC, Bangalore)
Chittoor – 517127

INSTITUTE VISION AND MISSION

INSTITUTE VISION

To emerge as a Centre of Excellence for Learning and Research in the domains of engineering, computing and management.

INSTITUTE MISSION

- Provide congenial academic ambience with state-art of resources for learning and research.
- Ignite the students to acquire self-reliance in the latest technologies.
- Unleash and encourage the innate potential and creativity of students.
- Inculcate confidence to face and experience new challenges.
- Foster enterprising spirit among students.
- Work collaboratively with technical Institutes / Universities / Industries of National and International repute

DEPARTMENT VISION AND MISSION

DEPARTMENT VISION

To become the Centre of excellence for skilled software professionals in Computer Applications.

DEPARTMENT MISSION

- Provide congenial academic ambiance with necessary infrastructure and learning resources.
- Inculcate confidence to face and experience new challenge from industry and society
- Ignite the students to acquire self reliance in the State-of-the Art Technologies.
- Foster Enterprise spirit among students

Post Graduates of Computer Applications shall

PEO1: Have Professional competency through the application of knowledge gained from fundamental and advanced concepts of structural and functional components in software. (Professional Competency)

PEO2: Excel in one's career by critical thinking toward successful services and growth of the organization or as an entrepreneur or through higher studies. (Successful Career Goals)

PEO3: Enhance Knowledge by updating advanced technological concepts for facing the rapidly changing world and contribute to society through innovation and creativity. (Continuing Education to Society)

PROGRAMME OUTCOMES (PO's)

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

SREENIVASA INSTITUTE OF TECHNOLOGY AND MANAGEMENT STUDIES.
(AUTONOMOUS)
MCA DEPARTMENT

I MCA - I Semester	**L**	**T**	**P**	**C**
	0	**0**	**3**	**1.5**

18MCA 115 PROGRAMMING FOR PROBLEM SOLVING LAB

PREREQUISITES: A Course on "Programming for Problem Solving"

Course Educational Objectives:

CEO1	To acquire knowledge about the basic concept of writing a C program.
CEO2	Know the role of constants, variables, identifiers, operators, type conversion and other building blocks of C Language.
CEO3	Use of conditional expressions and looping statements to solve problems associated with conditions and repetitions.
CEO4	Know the role of Functions involving the idea of modularity.
CEO5	Learn concept of Array and pointers dealing with memory management and files.

Syllabus:

Implement the Following by using C Language

1. To Calculate Area & Circumference of a Circle.
2. To Swap Two numbers With & Without using Temporary Variable.
3. To print the size of every Data type.
4. To Calculate Bill Amount for an item given its quantity sold, amount, discount & tax.
5. To find biggest among 3 numbers.
6. To find sum of first n numbers.
7. To find multiplication table for a given input value.
8. To generate Odd or Even number upto 100 and super number from 1000 to 9999.
9. To generate Fibonacci series for a given input.
10. To obtain sum of the first 10 terms of the following series for any Positive integer value of X: X +X3 /3! +X5/5! ! +X7/7! + …
11. To reverse the digits of a given number. For example, the number 9876 should be returned as 6789.
12. To remove duplicates from an ordered array. For example, if input array contains 10,10,10,30,40,40,50,80,80,100 then output should be 10,30,40,50,80.
13. Apply recursive call to do the following:

a) Find the factorial of a given number.

b) Compute $^{n}C_{r}$ value.

14. To convert uppercase string to lowercase string & Vice Versa without using string function.
15. To convert the two-dimensional array into one-dimensional array.
16. To find Binary Equivalent of a given number.
17. To display the Floyed's Triangle pattern.
18. To display different number pattern
19. To perform addition of two given matrices.
20. To perform multiplication of two given matrices.
21. To find the transpose of a given matrix.
22. To calculate Salary for 5 Employees using Structure.
23. To copy the content from one file to another file.
24. To count the number of vowels present in a file

Course Outcomes:

At the end of the course, students will be able to

COURSE OUTCOMES		
CO1	Demonstrate the knowledge on basic usage of operators, datatypes, variable declaration, looping & branching, arrays, strings, pointers, structures & union and files	PO1
CO2	Analyse & Develop an algorithm for every problem to be solved	PO2
CO3	Implement every program based logic involved in Algorithm	PO3
CO4	Test every program for different inputs to get effective solutions	PO4
CO5	Use appropriate software to implement program and to obtain solution	PO5
CO6	Relate programming principles to implement every program	PO8
CO7	Inspect every program individually for effective practice	PO9
CO8	The result and bugs of every program is observed and recorded in observation	PO10
CO9	Assess the technological changes in which it correlates to change and need	PO12

.

CO Vs PO Mapping

Course	PO / CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
C106 – Programming for Problem Solving	C106.1	3	-	-	-	-	-	-	-	-	-	-	-
	C106.2	-	3	-	-	-	-	-	-	-	-	-	-
	C106.3	-	-	3	-	-	-	-	-	-	-	-	-
	C106.4	-	-	-	3	-	-	-	-	-	-	-	-
	C106.5	-	-	-	-	3	-	-	-	-	-	-	-
	C106.6	-	-	-	-	-	-	-	3	-	-	-	-
	C106.7	-	-	-	-	-	-	-	-	3	-	-	-
	C106.8	-	-	-	-	-	-	-	-	-	3	-	-
	C106.9	-	-	-	-	-	-	-	-	-	-	-	3
	C106	3	3	3	3	3	-	-	3	3	3	-	3

TABLE 1: Rubrics for Programming for Problem Solving Lab

	Excellent(3)	Good(2)	Fair(1)
Conduct Experiments (CO1)	Student successfully completes the experiment, records the data, analyzes the experiment's main topics, and explains the experiment concisely and well.	Student successfully completes the experiment, records the data, and analyzes the experiment's main topics	Student successfully completes the experiment, records the data, and unable to analyzes.
Analysis and Synthesis (CO2)	Thorough analysis of program developed	Reasonable analysis of program developed	Improper analysis of program developed
Design (CO3)	Student understands what needs to be tested and designs an appropriate experiment, and explains the experiment concisely and well	Student understands what needs to be tested and designs an appropriate experiment.	Student understands what needs to be tested and does not design an appropriate experiment.
Complex Analysis & Conclusion (CO4)	Thorough comprehension through analysis/ synthesis	Reasonable comprehension through analysis/ synthesis	Improper comprehension through analysis/ synthesis
Use modern tools in executing the programs (CO5)	Student uses the tools to develop and execute the programs, and understands the limitations of the tool.	Student uses the tools correctly.	Student uses the tools correctly, unable to understand properly.
Report Writing (CO6)	Status report with clear and logical sequence of parameter using excellent language	Status report with logical sequence of parameter using understandable language	Status report not properly organized
Lab safety (CO7)	Student will demonstrate good understanding and follow lab safety	Student will demonstrate good understanding of lab safety	Students demonstrate a little knowledge of lab safety.
Ability to work in teams (CO8)	Performance on teams is excellent with clear evidence of equal distribution of tasks and Effort	Performance on teams is good with equal distribution of tasks and effort	Performance on teams is acceptable with one or more members carrying a larger amount of the effort
Continuous learning (CO9)	Highly enthusiastic towards continuous learning	Interested in continuous learning	Inadequate interest in continuous learning

Course Outcome Attainment (R18)

Day – To – Day Evaluation	**Fair**	Level 1	If Student scored less than 80% of the total mark allotted.
	Good	Level 2	If Student scored greater than 80 % and less than 90% of the total mark allotted.
	Excellent	Level 3	If Student scored greater than 90% of the total mark allotted.
Term End Exam (TEE)	**Fair**	Level 1	If Student scored less than 80% of the total mark allotted.
	Good	Level 2	If Student scored greater than 80 % and less than 90% of the total mark allotted.
	Excellent	Level 3	If Student scored greater than 90% of the total mark allotted.

SITAMS	PROGRAMMING FOR PROBLEM SOLVING LAB	18MCA115

Sl. No.	Date	Name of the Exercise	Page No.	Signature
1		To Calculate Area & Circumference of a Circle.		
2		To Swap Two numbers With & Without using Temporary Variable		
3		To print the size of every Data type.		
4		To Calculate Bill Amount for an item given its quantity sold, amount, discount & tax		
5		To find biggest among 3 numbers.		
6		To find sum of first n numbers.		
7		To find multiplication table for a given input value.		
8		To generate Odd or Even number upto 100 and super number from 1000 to 9999.		
9		To generate Fibonacci series for a given input.		
10		To obtain sum of the first 10 terms of the following series for any Positive integer value of X: X +X3 /3! +X5/5! ! +X7/7! + …		
11		To reverse the digits of a given number. For example, the number 9876 should be returned as 6789.		
12		To remove duplicates from an ordered array. For example, if input array contains 10,10,10,30,40,40,50,80,80,100 then output should be 10,30,40,50,80.		
13		Apply recursive call to do the following: Find the factorial of a given number. Compute ${}^{n}C_{r}$ value.		
14		To convert uppercase string to lowercase string & Vice Versa without using string function.		
15		To convert the two-dimensional array into one-dimensional array.		
16		To find Binary Equivalent of a given number		
17		To display the Floyd's Triangle pattern		
18		To display different number pattern		
19		To perform addition of two given matrices.		
20		To perform multiplication of two given Matrices.		
21		To find the transpose of a given matrix.		

22		To calculate Salary for 5 Employees using Structure.		
23		To copy the content from one file to another file.		
24		To count the number of vowels present in a file		

SREENIVASA INSTITUTE OF TECHNOLOGY AND MANAGEMENT STUDIES (AUTONOMOUS)
Chittoor – 517127

MCA DEPARTMENT
Table2: INDEX SHEET

Name: **Roll No.** **Year & Sem : I / I** **AY:**

S.No	Exercise Name	Knowledge Gained	Analysis, Design and Use of Modern Tool / Technique	Ability of do experiment and following of ethical principles	Result & Conclusion	VIVA VOCE (Communication, Life Long learning	TOTAL	Signature of the Faculty
		5	10	10	10	5	40M	
1	To Calculate Area & Circumference of a Circle. To Swap Two numbers With & Without using Temporary Variable. To print the size of every Data type.							
2	To Calculate Bill Amount for an item given its quantity sold, amount, discount & tax. To find biggest among 3 numbers. To find sum of first n numbers.							
3	To find multiplication table for a given input value. To generate Odd or Even number upto 100 and super number from 1000 to 9999.							
4	To generate Fibonacci series for a given input. To obtain sum of the first 10 terms of the following series for any Positive integer value of X: X +X3 /3! +X5/5! ! +X7/7! + …							
5	To reverse the digits of a given number. For example, the number 9876 should be returned as 6789. To remove duplicates from an ordered array. For example, if input array contains 10,10,10,30,40,40,50,80,80,100 then output should be 10,30,40,50,80.							

6	Apply recursive call to do the following: Find the factorial of a given number. Compute $^{n}C_{r}$ value.							
7	To convert uppercase string to lowercase string & Vice Versa without using string function. To convert the two-dimensional array into one-dimensional array.							
8	To find Binary Equivalent of a given number. To display the Floyd's Triangle pattern							
9	To display different number pattern							
10	To perform addition of two given matrices. To perform multiplication of two given matrices.							
11	To find the transpose of a given matrix. To calculate Salary for 5 Employees using Structure.							
12	To copy the content from one file to another file. To count the number of vowels present in a file							

SREENIVASA INSTITUTE OF TECHNOLOGY AND MANAGEMENT STUDIES
(AUTONOMOUS)
Chittoor – 517127

MCA DEPARTMENT

Table 3: ATTAINMENT SHEET

Name: **Roll No.** **Year & Sem : I / I** **AY:**

S.No	Exercise Name	CO1	CO2	CO3	CO4	CO5	CO6	CO7	CO8	CO9
		Knowledge	Analysis	Design	Complex Analysis & Conclusion	Use of modern tools	Communication ability	Ethics	Individual / Team work	Life Long Learning
1	To Calculate Area & Circumference of a Circle. To Swap Two numbers With & Without using Temporary Variable. To print the size of every Data type.									
2	To Calculate Bill Amount for an item given its quantity sold, amount, discount & tax. To find biggest among 3 numbers. To find sum of first n numbers.									
3	To find multiplication table for a given input value. To generate Odd or Even number upto 100 and super number from 1000 to 9999.									
4	To generate Fibonacci series for a given input. To obtain sum of the first 10 terms of the following series for any Positive integer value of X: X +X3 /3! +X5/5! ! +X7/7! + …									

5	To reverse the digits of a given number. For example, the number 9876 should be returned as 6789. To remove duplicates from an ordered array. For example, if input array contains 10,10,10,30,40,40,50,80,80,100 then output should be 10,30,40,50,80.									
6	Apply recursive call to do the following: Find the factorial of a given number. Compute $^{n}C_{r}$ value.									
7	To convert uppercase string to lowercase string & Vice Versa without using string function. To convert the two-dimensional array into one-dimensional array.									
8	To find Binary Equivalent of a given number. To display the Floyd's Triangle pattern									
9	To display different number pattern									
10	To perform addition of two given matrices. To perform multiplication of two given matrices.									
11	To find the transpose of a given matrix. To calculate Salary for 5 Employees using Structure.									
12	To copy the content from one file to another file. To count the number of vowels present in a file									
Average of Day– to – Day evaluation (C1)										

Signature of the Faculty

Exercise 1

CALCULATE AREA & CIRCUMFERENCE OF THE CIRCL

Aim

To implement a C Program to calculate Area and Circumference of the circle

Algorithm

```
Step 1: Start
Step 2: Declare the float variables pi and assign the value of pi ←3.14, area and  cir
Step 3: Declare the integer variable ‘radius’
Step 4: Read the value for ‘radius’
Step 5: Compute area←pi*radius*radius
Step 6: Compute cir←2*pi*radius
Step 7: Print area, cir
Step 8: Stop
```

Exercise 2

SWAP TWO NUMBERS WITH & WITHOUT USING TEMPORARY VARIABLES

Aim

To implement a C Program to swap two numbers with & without using temporary variables

Algorithm

```
Step 1: Start
Step 2: Declare the integer variable a,b,c,d,t
Step 3: Read the value for a & b
Step 4: Print "Before swapping"
Step 5: Print a,b
Step 6: Assign t←a
Step 7: Assign a←b
Step 8: Assign b←t
Step 9: Print "After swapping with using temporary variables"
Step 10: Print a,b
Step 11: Assign the value c←10,d←20
Step 12: Print "Before swapping"
Step 13: Print c,d
Step 14: Compute c←(c+d)-(d=c)
Step 15: Print "After swapping without using temporary variables"
Step 16: Print c,d
Step 17: Stop
```

Exercise 3

SHOW THE SIZE OF THE DATATYPES

Aim

To implement a C Program to show the size of the datatypes

Algorithm

```
Step 1: Start
Step 2: Declare the array variables belongs to data type in integer a[1..5],
          float b[1.5], char c[1..5],double d[1..5]
Step 3: Print sizeOf(a)
Step 4: Print sizeOf(b)
Step 5: Print sizeOf(c)
Step 6: Print sizeOf(d)
Step 7: Stop
```

Exercise 4

CALCULATE BILL AMOUNT FOR AN ITEM GIVEN ITS QUANTITY SOLD, AMOUNT, DISCOUNT & TAX.

Aim

To implement a C Program to calculate bill amount for an item given its quantity sold, amount, discount and tax

Algorithm

```
Step 1: Start
Step 2: Declare  variables as a double quantity,amount,price,discount
Step 3: Read quantity and price.
Step 4: amount=quantity*price
Step 4.1: if(amount>5000)
Step 4.1.1: discount=amount*0.05
Step 4.1.2: amount=amount-discount
Step 5: Print amount
Step 6: Stop
```

Exercise 5

GREATEST NUMBER AMONG THREE NUMBERS

Aim:

To implement C program to find out biggest number among three numbers.

Algorithm:

```
Step 1  :  Start
Step 2  :  Declare the variable a,b,c
Step 3  :  Read the value for variable 'a' & 'b'
Step 4  :  Compare a>b && a>c
    4a    :  Print 'a' is biggest number
     4.1  :  Compare b>a && b>c
    4.1.a :  Print 'b' is biggest number
    4.1.b    Print 'c' is biggest number
Step 5  :  Stop
```

Exercise 6

SUM OF 'N' NUMBERS

Aim:

To implement C program to find the sum of 'N' natural numbers.

Algorithm:

```
Step 1 : Start
Step 2 : Declare the variable sum, n
Step 3 : Assign the variable sum ←0
Step 4 : Read the value for variable 'n'
Step 5 : Compute sum←n*(n-1)/2
Step 6 : Print sum
Step 7 : Stop
```

Exercise 7

MULTIPLICATION TABLE GENERATION

Aim

To implement C program to generate multiplication tables for given table number.

Algorithm

```
Step 1 :  Start
Step 2 :  Declare the variable i ,tv
Step 3 :  Read the value for the variable 'tv'
Step 4 :  Assign the variable i←1
Step 5 :  Print "the multiplication table n"
Step 6 :  Repeat i until i<=10
     6.1 :  Print i, tv, i*tv
Step 7 :  Stop
```

Exercise 8

GENERATE ODD, EVEN AND SUPER NUMBER

Aim

To implement C program to generate odd, even and super number.

Algorithm

```
Step 1: Start
Step 2: Declare the integer variable I,q,r,m,n
Step 3: Print "The even numbers are"
Step 4: Assign the value i←0
Step 5: Repeat i until i<=100
Step 5.1: if(i mod 2==0)
Step 5.1.1: Print i
Step 5.2: Compute i←i+1
Step 6: Print "The odd numbers are"
Step 7: Assign the value i←0
Step 8: Repeat i until i<=100
Step 8.1: if(i mod 2 !=0)
Step 8.1.1: Print i
Step 8.2: Compute i←i+1
Step 9: Print "Super number are:"
Step 10: Assign the value i←1000
Step 11: Repeat i until i<=9999
Step 11.1: Compute q←i/100
Step 11.2: Compute r←imod100
Step 11.3:Compute m←q+r
Step 11.4: Compute n←m*m
Step 11.5: if(n==i)
Step 11.5.1: Print i
Step 11.6: Compute i←i+1
Step 12: Stop
```

Exercise 9

TO GENERATE FIBONACCI SERIES FOR A GIVEN INPUT.

Aim

To implement C program to generate Fibonacci series for a given input

Algorithm

```
Step 1: Start
Step 2: Declare the integer variable f1,f2,f3,count,n
Step 3: f1<-1, f2<- 1,count<-1
SteP 4: Read the value for variable 'n'
Step 5: Print "The Fibonacci series are"
SteP 6: Repeat count until count<=n
        6.1: Compute f3<-f1+f2
        6.2: Print f3
        6.3: f1<-f2
        6.4: f2<-f3
        6.5: Compute count<-count+1
Step 7: Stop
```

Exercise 10

SUM OF 'N' TERMS

Aim

To implement C program to calculate the sum of 'n' terms.

Algorithm

```
Step 1: Start
step 2: Declare the double variable x,sum
Step 3: Declare integer variable n,i,j
Step 4: Declare the long integer variable fact
Step 5: Sum<-0.00,i<-1,j<-1
Step 6: Read the value of 'x'
Step 7: Read the value of 'n'
Step 8: Repeat i until i<=n
        8.1: fact<-1
        8.2: Repeat j until j<=i
                8.2.1: fact<-fact*j
                8.2.2: j<-j+1
        8.3: Compute sum<-sum+(pow(x,i)/fact)
        8.4: i<-i+2
Step 9: Print sum
Step 10: Stop
```

Exercise 11

REVERSE OF A GIVEN 4 DIGIT NUMBER

Aim

To implement C program to reverse the given four digit number.

Algorithm

```
Step 1 :  Start
Step 2  :  Declare the variable 'n,d,rev'
Step 3  :  Read the value for variable 'n'
Step 4  :  Assign the variable rev←0
Step 5  :  Repeat step 5.1 to 5.3 until n!=0
     5.1  :  Compute d←n mod 10
     5.2  :  Compute rev←rev+d*10
     5.3  :  Compute n←n/10
Step 6  :  Print 'rev'
Step 7  :  Stop
```

Exercise 12

DUPLICATE VALUE REMOVAL FROM ARRAY

Aim

To implement C program to remove duplication from an ordered array.

Algorithm

```
Step 1  :  Start
Step 2  :  Declare the integer variable i, j, k, n
Step 3  :  Declare and assign the integer variable
              arr[ ]={10,10,10,30,40,40,50,80,80,100}
Step 4  :  Compute n←sizeof(arr)/ sizeof(arr[0])
Step 5  :  Assign the variable i←0, j←i+1
Step 6  :  Repeat i until i<n
     6.1  :  Repeat j until j<n
     6.1.1  : if (arr[j] equals arr[i])
     6.1.2  : k←j
     6.1.3  : Repeat k until k<n
      6.1.3.1 :  arr[k]←arr[k+1]
     6.1.4  :  Compute n←n-1
     6.1.5  :  else compute j←j+1
Step 7  :  Assign the variable i←0
Step 8  :  Repeat i until i<n
Step 9  :  Print a
rr[j]
Step 10 : Stop
```

Exercise 13

FACTORIAL NUMBER USING RECURSIVE FUNCTION

Aim

To implement C program to find factorial of given number using recursive function.

Algorithm

```
Step 1 : Start
Step 2 : Declare the integer variable x, f
Step 3 : Declare the function fact(int) with return type integer
Step 4 : Read the value for the variable 'x'
Step 5 : Compute f←fact(x)
Step 6 : Print f
Step 7 : Stop
```

Fact( ):

```
Step 1 : Start
Step 2 : Declare the integer variable f, m
Step 3 : f←1
Step 4 : if (m equals 1)
    4.1 : return 1
    4.2 : else
    4.3 : Compute f←m*fact(m-1)
    4.4 : return f
Step 5 : Stop
```

^nC_r Calculation

Aim

To implement C program to compute ^nC_r value.

Algorithm

```
Step 1 : Start
Step 2 : Declare the integer variable p,r,n
Step 3 : Read the value for the variable 'n' & 'r'
Step 4 : p←fact(n)/(fact(n-r)*fact(r))
Step 5 : Print n,r,p
Step 6 : Stop
Algorithm for function definition fact(c):
Step 1 : Start
Step 2 : Declare integer variable c, n
Step 3 : Assign the variable f←1
Step 4 : if (c greater than 0)
     4.1 : Compute f←f*c
     4.2 : c←c-1
Step 5 : return f
Step 6 : Stop
```

Exercise 14

CASE CONVERSION OF STRING

Aim

To implement C program to convert given string from upper case to lower case and vice versa without using string function.

Algorithm

```
Step 1 : Start
Step 2 : Declare integer variable i and character variables l[1..15],ch
Step 3 : Read the value for the variable "ch"
Step 4 : if (ch equals 'l')
    4.1 : Read the lower case string for the variable 'l'
    4.2 : Assign the variable i←0
    4.3 : Repeat i until l[i]!='\0'
    4.4 : Print toascii ((l[i])-32)
    4.5 : Compute i←i+1
Step 5 : if (ch equals 'u')
    5.1 : Read the upper case string for the variable 'l'
    5.2 : Assign the variable i←0
    5.3 : Repeat i until l[i]!='\0'
    5.4 : Print toascii ((l[i])+32)
    5.5 : Compute i←i+1
Step 6 : Stop
```

Exercise 15

CONVERSION OF TWO DIMENSIONAL ARRAY IN TO ONE DIMENSIONAL ARRAY

Aim

To implement C program to convert two dimensional array to one dimensional array.

Algorithm

```
Step 1 : Start
Step 2 : Declare the variable i, j, k, a[1..2][1..3],b[1..6]
Step 3 : Assign the variable a[ ][ ]←{{2,3,4},{5,6,7}}, i←0, j←0
Step 4 : Repeat i until i<2
    4.1 : Repeat j until j<3
    4.1.1 : Assign b[k]←a[i][j]
    4.1.2 : Print one dimensional array k, b[k]
    4.1.3 : Compute k←k+1
    4.1.4 : Compute j←j+1
    4.2 : Compute i←i+1
Step 5 : Stop
```

Exercise 16

DECIMAL TO BINARY CONVERSION

Aim

To implement C program to convert binary equivalent to a given number.

Algorithm

```
Step 1 :  Start
Step 2 :  Declare the int variable a[1..20], dec, i, j, r
Step 3 :  Assign the variable i←0
Step 4 :  Read the value for the variable 'dec'
Step 5 :  Assign the value r←dec
Step 6 :  Repeat r until r>0
     6.1 :  Compute a[i]←r mod 2
     6.2 :  Compute i←i+1
     6.3 :  Compute r←r/2
Step 7 :  Repeat step 7.1 to 7.2 until j>=0
     7.1 :  Print a[j]
     7.2 :  j←j-1
Step 8 :  Stop
```

Exercise 17

FLOYD TRIANGLE NUMBER PATTERN

Aim

To implement C program to generate Floyd triangle number pattern.

Algorithm

```
Step 1 :  Start
Step 2 :  Declare the integer variable i, j, n
Step 3 :  Assign the variables n←1, j←1, i←1
Step 4 :  Repeat i until i<=5
      4.1 :  Repeat j until j<=1
   4.1.1 :  Print n
  4.1.2  :  Compute j←j+1
     4.2  :  Print "\n"
     4.3  :  Compute i←i+1
Step 5 :  Stop
```

Exercise 18

NUMBER PATTER GENERATION 1

Aim

To implement C program to generate number pattern.

Algorithm

```
Step 1 : Start
Step 2 : Declare the variable i, j
Step 3 : Assign the variable i←1, j←1
Step 4 : Repeat i until i<=5
    4.1 : Repeat j until j<=i
    4.2 : Compute j←j+1
    4.3 : Compute i←i+1
    4.4 : Print 'i'
    4.5 : Print "\n"
Step 5 : Stop
```

NUMBER PATTER GENERATION 2

Aim

To implement C program to generate number pattern.

Algorithm

```
Step 1 : Start
Step 2 : Declare the integer variable i, j
Step 3 : Assign the variable i←1, j←1
Step 4 : Repeat i until i<=5
    4.1 : Repeat j until j<=i
    4.2 : Print j
    4.3 : j←j+1
    4.4 : Print "\n"
    4.5 : i←i+1
Step 5 : Stop
```

NUMBER PATTERN GENERATION 3

Aim

To implement C program to generate reverse number pattern.

Algorithm

```
Step 1  :  Start
Step 2  :  Declare the integer variable i, j
Step 3  :  Assign the variables i←5, j←1
Step 4  :  Repeat i until i>=1
     4.1  :  Repeat j until j<=1
     4.2  :  Print j
     4.3  :  Compute j←j+1
     4.4  :  Print "\n"
     4.5   :  Compute i←i-1
Step 5  :  Stop
```

Exercise 19

ADDITION OF TWO MATRICES

Aim

To implement C program to add two matrices.

Algorithm

```
Step 1  : Start
Step 2  : Declare the integer variable a[1. .3][1. .3], b[1. .3][1. .3],
              c[1. .3][1. .3], i, j
Step 3  : Assign the variable i←0, j←0
Step 4  : Repeat i until i<2
     4.1  : Repeat j until j<2
   4.1.1  : Read the value for the variable a[i][j]
   4.1.2  : j←j+1
     4.2  : i←i+1
Step 5  : Assign the variable i←0, j←0
Step 6  : Repeat i until i<2
     6.1  : Repeat j until j<2
     6.1.1  : Read the value for the variable b[i][j]
     6.1.2   : j←j+1
     6.2   : i←i+1
Step 7  :  Print "matrix addition"
Step 8  : Assign the variable i←0, j←0
Step 9  : Repeat i until i<2
      9.1  : Repeat j until j<2
      9.1.1   : Compute c[i][j]←a[i][j]+b[i][j]
      9.1.2   : j←j+1
      9.2   : i←i+1
Step 10 : Assign the variable i←0, j←0
Step 11 : Repeat i until i<2
       11.1 : Repeat j until j<2
       11.1.1  : Print c[i][j]
       11.1.2  : j←j+1
       11.2  : i←i+1
Step 12 : Stop
```

Exercise 20

MULTIPLICATION OF TWO MATRICES

Aim

To implement C program to multiply two matrices.

Algorithm

```
Step 1: Start
Step 2: Declare the integer variable a[1..5][1..5], b[1..5][1..5], c[1..5][1..5],i,j,k,r1,c1
Step 3: Read the value for variable r1 and c1
Step 4: i<-0, j<-0
Step 5: Repeat i until i<r1
        5.1: Repeat j until j<c1
             5.1.1: Read the value for variable a[i][j]
             5.1.2: j<-j+1
             5.2: i<-i+1
Step 6: i<-0, j<-0
Step 7: Repeat i until i<r1
        7.1: Repeat j until j<c1
             7.1.1: Read the value for variable b[i][j]
             7.1.2: j<-j+1
             7.2: i<-i+1
Step 8: Print "Matrix Multiplication"
Step 9: i<-0, j<-0, k<-0
Step 10: Repeat i until i<r1
        10.1: Repeat j until j<c1
        10.1.1: c[i][j]<-0
        10.1.2: Repeat k until k<c1
        10.1.2.1: Compute c[i][j]<-c[i][j]+a[i][k]*b[k][j]
        10.1.2.2: k<-k+1
        10.2: print c[i][j]
        10.3: j<-j+1
        10.4: i<-i+1
Step 11: Stop
```

Exercise 21

TRANSPOSE OF MATRIX

Aim

To implement C program to transpose a given matrix.

Algorithm

```
Step 1 :  Start
Step 2 :  Declare the integer variable a[1. .10][1. .10], i, j,m,n
Step 3 :  Read the value for the variable m,n
Step 4 :  i←0, j←0
Step 5 :  Repeat i until i<m
     5.1 :  Repeat j until j<n
Step 6 :  Assign the variables i←0, j←0
Step 7 :  Print "The given matrix is"
Step 8 :  Repeat i until i<m
     8.1 :  Repeat j until j<n
     8.2 :  Print a[i][j]
     8.3 :  Print "\n"
Step 9 :  Print "Transpose of given matrix is"
Step 10:  i←0, j←0
Step 11:  Repeat j until j<n
     11.1:  Repeat i until i<m
     11.2:  Print a[i][j]
     11.3:  Print "\n"
Step 12:  Stop
```

Exercise 22

EMPLOYEE SALARY USING ARRAY OF STRUCTURE

Aim

To implement C program to calculate employee salary using array of structure.

Algorithm

```
Step 1 :  Start
Step 2 :  Declare the integer variable i
Step 3 :  Define the structure with tag name 'compute' along with
          Internal members.
Step 4 :  Declare the character variable name[1. .10], integer variable
          Basic, hrap, float variable, hra & total
Step 5 :  Define structure variable sal[1. .2]
Step 6 :  i←1
Step 7 :  Repeat i until i<=2
     7.1 :  Read the value for the variable
          Sal[i].name, sal[i].basic, sal[i].hrap
     7.2 :  Compute sal[i].hra←sal[i].basic*sal[i].hrap/100
     7.3 :  Compute sal[i].total←sal[i].basic+sal[i].hra
     7.4 :  Compute i←i+1
Step 8 :  Assign the variable i←1
Step 9 :  Repeat i until i<=2
       9.1   :  Print sal[i].name, sal[i].basic, sal[i].hrap, sal[i].hra, sal[i].total
       9.2   :  Print "\n"
       9.3   :  i←i+1
Step10: Stop
```

Exercise 23

COPY THE CONTENT FROM ONE FILE TO ANOTHER FILE

Aim

To implement C program to copy the content from one file to another file

Algorithm

```
Step 1: Start
Step 2: Declare the file pointer variable '*fip','*fop' and integer variable  'c'.
Step 3: Open the text file with read mode and assign to 'fip'.
Step 4: Open the text file with write mode and assign to 'fop'.
Step 5: Check whether 'fip' exist or not.
Step 5.1: If not exist, program terminated.
Step 6: Read the content from 'fip' character by character until EOF.
Step 6.1: Write the content to 'fop' character by character.
Step 7: Close fip and fop.
Step 8: Stop
```

Exercise 24

COUNT THE NUMBER OF BLANK SPACE IN THE CONTENT OF THE FILE

Aim

To implement C program to count the number of blank space in the content of the file

Algorithm

```
Step 1: Start
Step 2: Declare the file pointer variable 'fp' and character variable  c and integer variable count
Step 3: Read file name
Step 4: Open the text file with read mode and assign to 'fp'
Step 5: Print "Read the content of the file"
Step 6: while((c=fgetc(fp))!=EOF)
Step 6.1: if(c==' ')
Step 6.1.1 compute count<- count + 1
Step 7: Print "the number of blank space is" count
Step 8: Close fp
Step 9: Stop
```

ADDITIONAL EXERCISE

Write the algorithm and implement the same by using C for the following problems

1. Accept any 5 digit number and print the remainder after dividing it by 3
2. Accept the marks for 5 subjects and calculate the percentage obtained
3. Accept any number n and print the cube of all number from 1 to n which is divisible by 3
4. Accept the money value and print how many notes from each denomination has to be fetched
5. Find out the roots of quadratic equation
6. Print given number in words using recursion. If number is 234, it prints two three four
7. Accept n numbers and display the sum of highest and lowest number
8. Accept m x n matrix and count the occurrence of a number in matrix
9. Declare the pointer variable for all datatypes and check the number of address allocated for each pointer variable
10. Accept any string, Reverse the string , find the length of the string with out using predefined function
11. Accept name and arrival time of five trains and display the name with rail time format