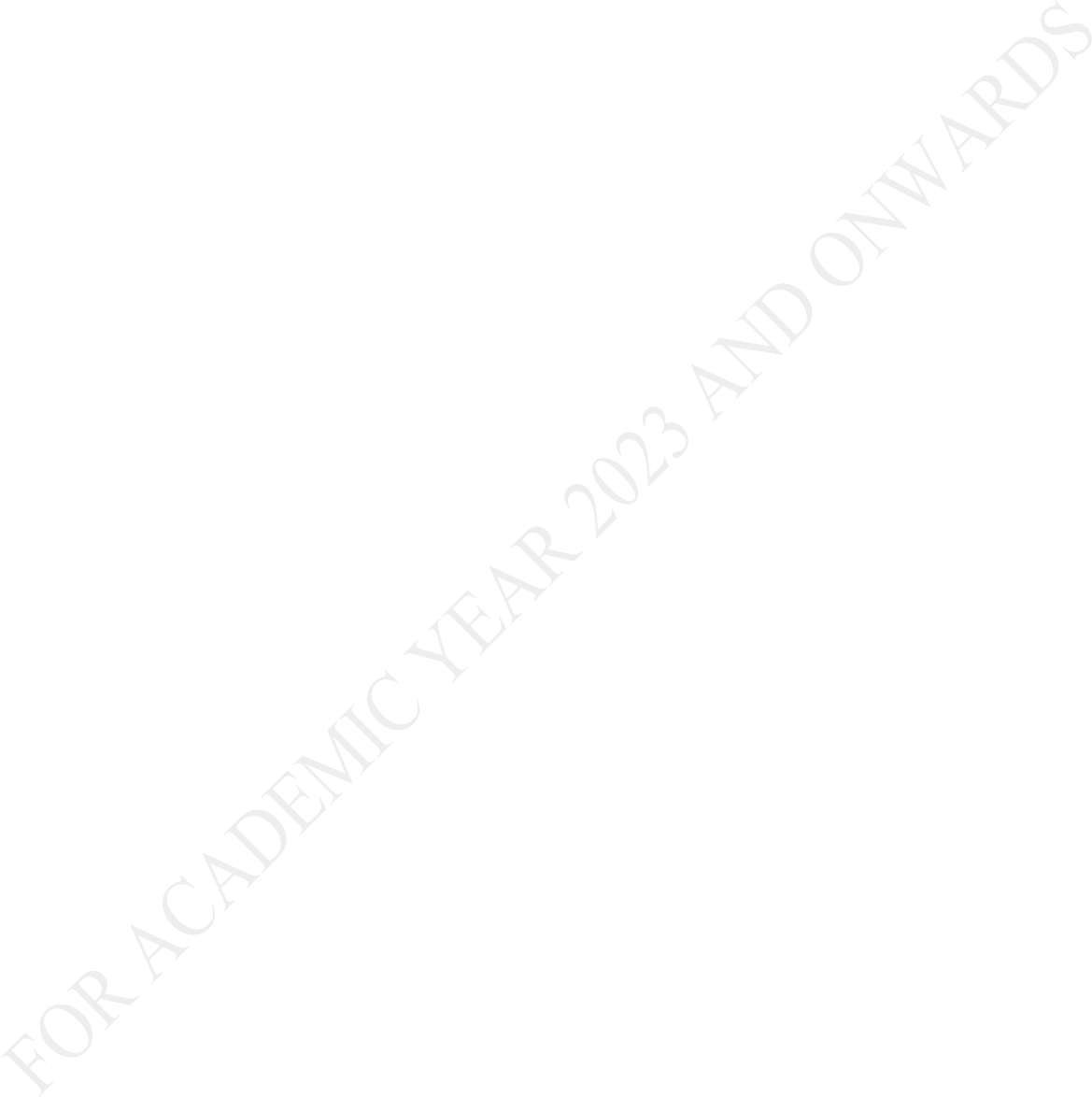


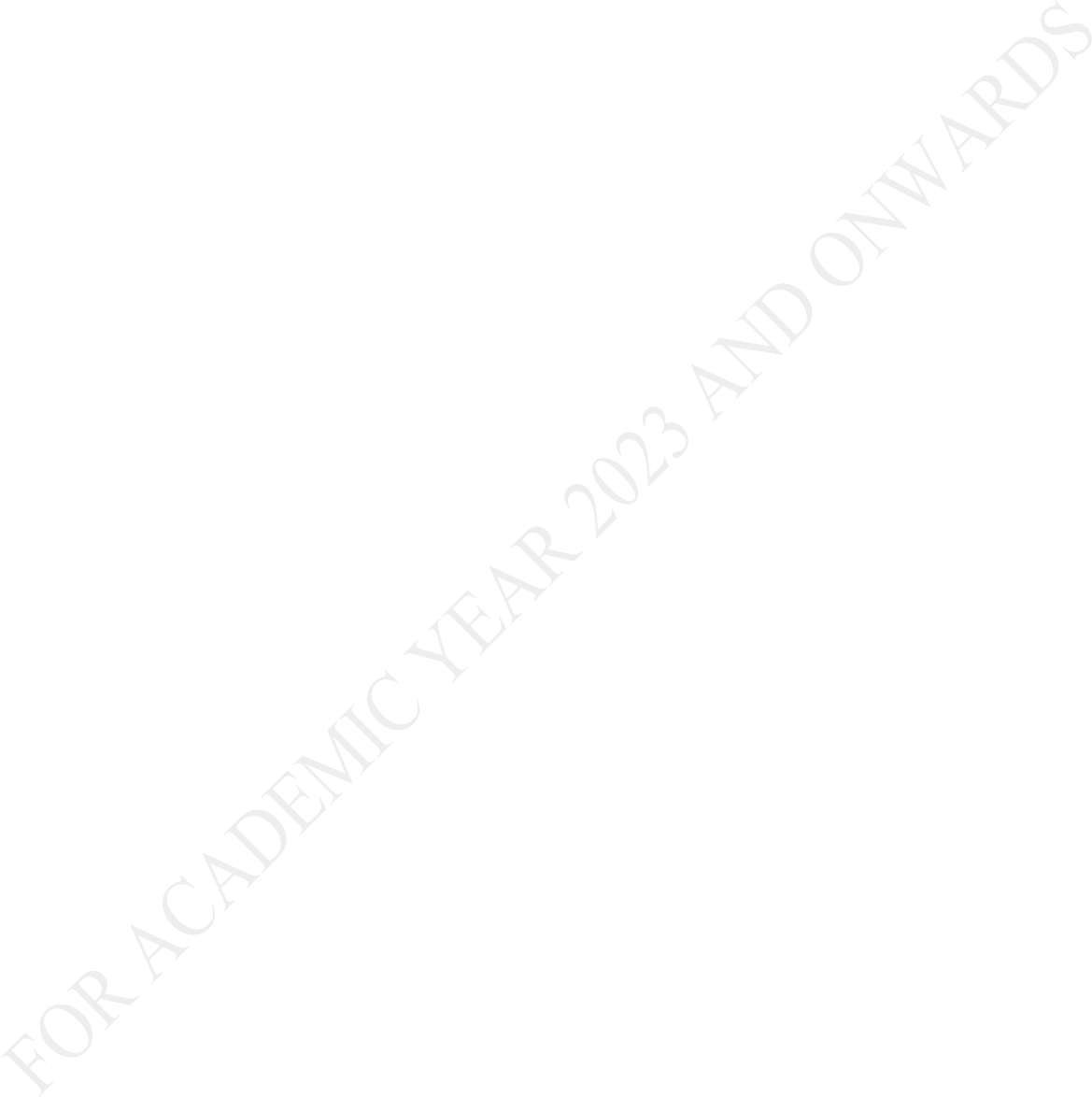
# Part II (Class XII)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Topics and Sub-topics** | **Student Learning Outcomes** | | **Cognitive Level** | | |
| **K** | **U** | **A** |
| **9. Operating System (OS)** | Students should be able to: | | | | |
| 9.1 Introduction to Operating System | 9.1.1 | identify the commonly used operating systems; | \* | \* |  |
| 9.1.2 | list the tasks performed by an operating system; |  |
| 9.1.3 | differentiate between command line interface and graphical  user interface of an operating system; | \* |
| 9.1.4 | differentiate between single user operating system and multi  user operating system; | \* |
| 9.1.5 | compare the types of operating system, i.e: | \* |
|  | a. simple batch system |  |
|  | b. multiprogramming batch system |  |
|  | c. multitasking operating system |  |
|  | d. distributed operating system |  |
|  | e. real-time operating system |  |
|  | f. parallel processing operating system  g. multiprocessor operating system |  |
|  | h. embedded operating system |  |
|  | i. time-sharing operating system; |  |
| 9.2 Functions of Operating System | 9.2.1 | describe the main functions of operating system, i.e. process management, memory management, file management, I/ O system management, secondary storage management, protection system, interrupt handling, network management, command-interpreter; |  | \* |  |
| 9.3 Process Management | 9.3.1 | determine the sequence of execution of processes to get the minimum execution time; |  |  | \* |
| 9.3.2 | explain the process state diagram including new, running,  waiting/ blocked, ready and terminated states of a process; | \* |
| 9.3.3 | differentiate between thread and process; | \* |

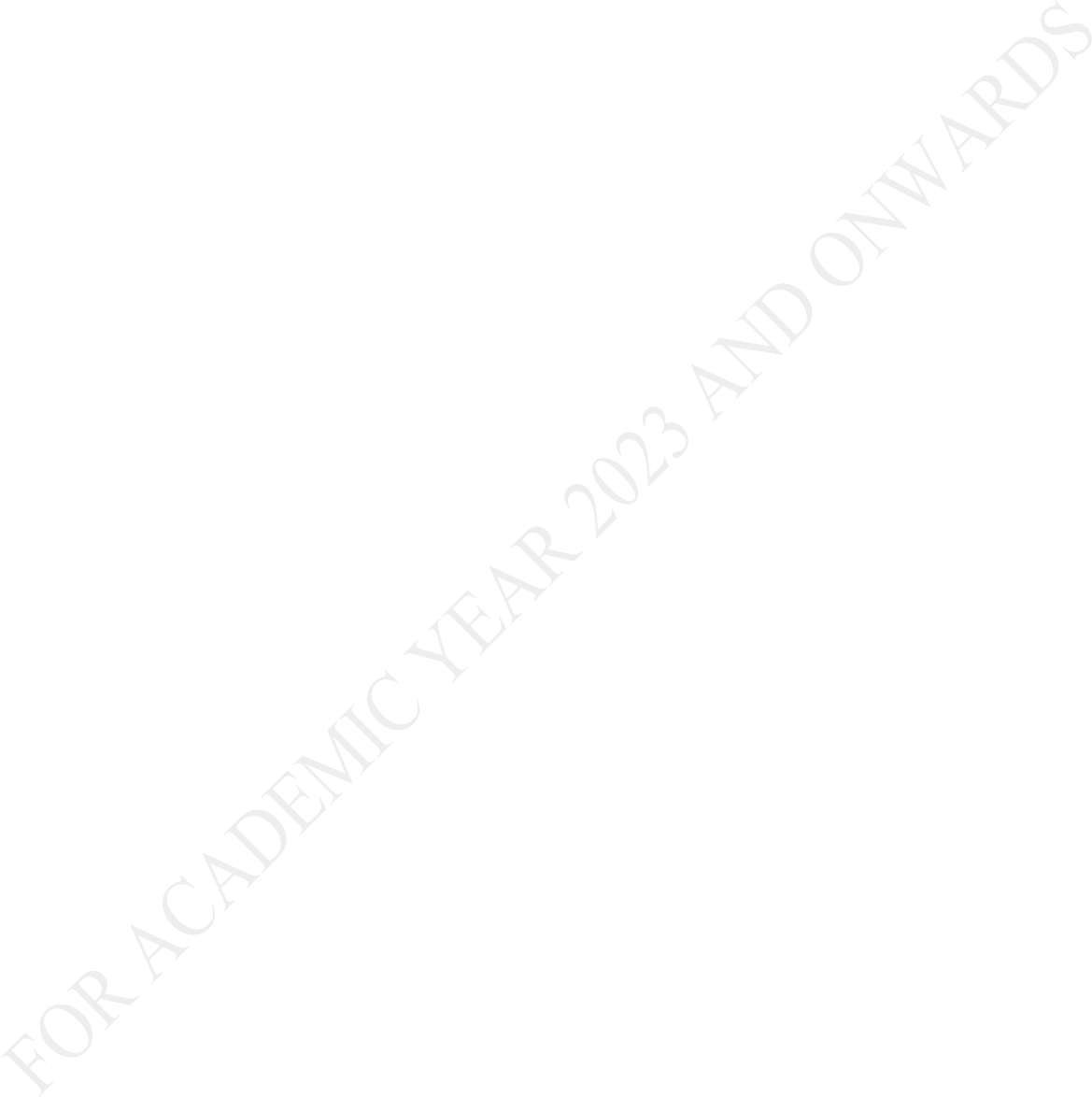
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Topics and Sub-topics** | **Student Learning Outcomes** | | **Cognitive Level** | | |
| **K** | **U** | **A** |
|  | Students should be able to: | | | | |
|  | 9.3.4 | differentiate among multithreading, multitasking, multiprogramming and multiprocessing. |  | \* |  |



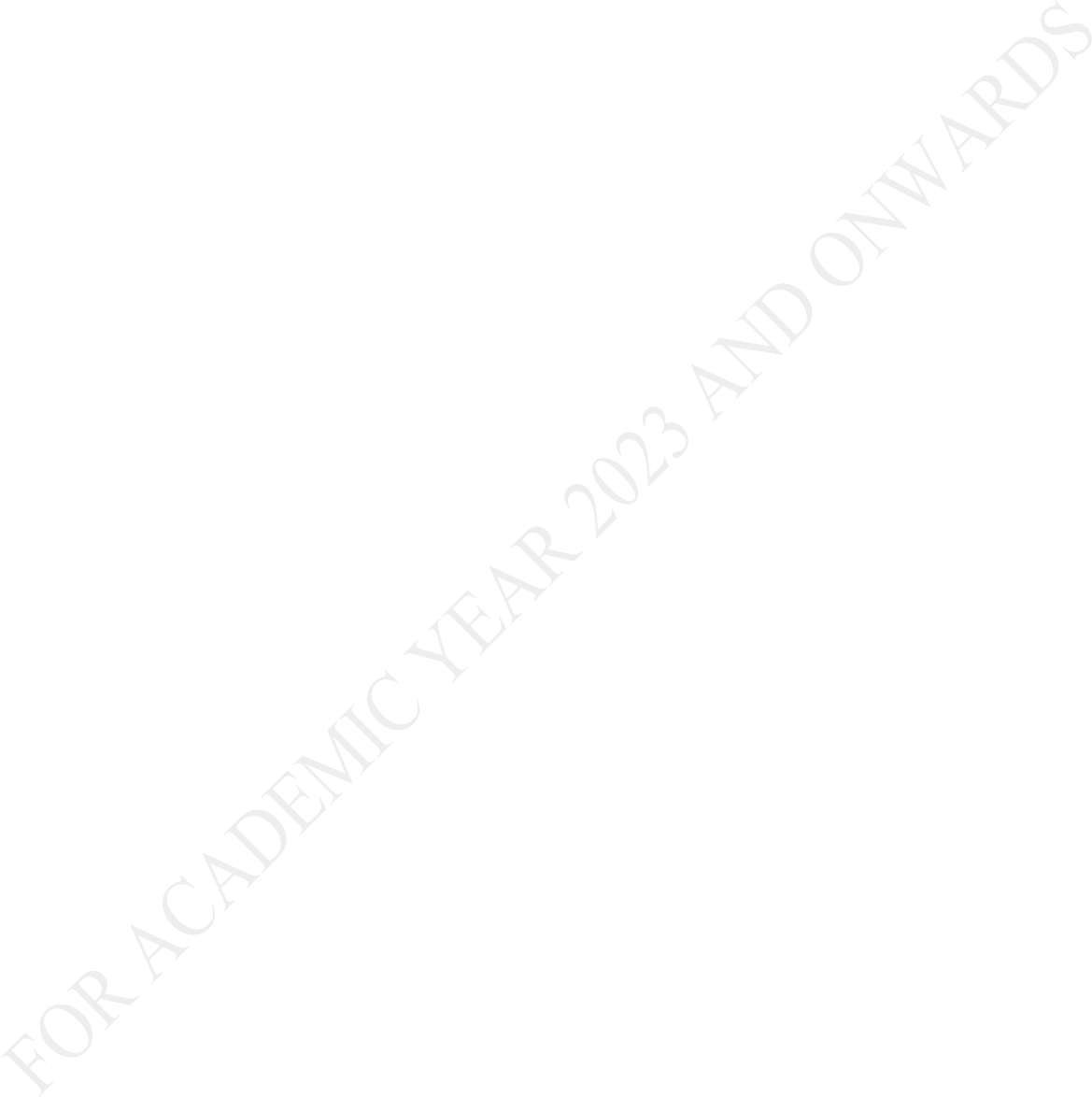
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Topics and Sub-topics** | **Student Learning Outcomes** | | **Cognitive Level** | | |
| **K** | **U** | **A** |
| **10. System Development Life Cycle**  **(SDLC)** | Students should be able to: | | | | |
| 10.1 System Development Life Cycle (SDLC) | 10.1.1 | define system; | \* |  |  |
| 10.1.2 | describe objectives of SDLC; | \* |
| 10.1.3 | describe the steps/ phases in SDLC; | \* |
| 10.1.4 | describe the responsibilities of the following personnel: | \* |
|  | a. management team |  |
|  | b. system analyst |  |
|  | c. project manager |  |
|  | d. programmer |  |
|  | e. software tester |  |
|  | f. customer. |  |



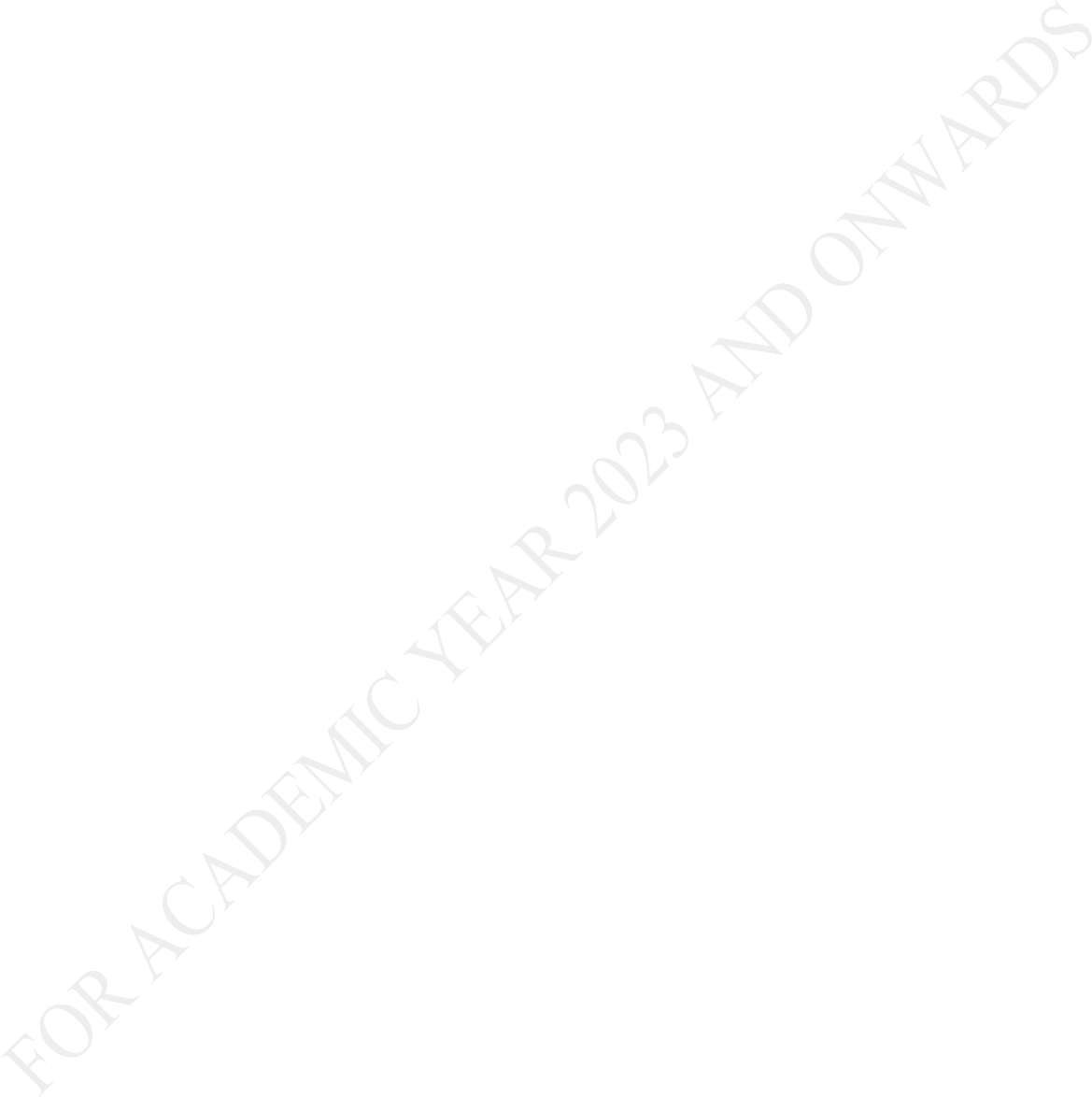
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Topics and Sub-topics** | **Student Learning Outcomes** | | **Cognitive Level** | | |
| **K** | **U** | **A** |
| **11. Introduction to C++ Programming** | Students should be able to: | | | | |
| 11.1 Programming Basics | 11.1.1 | define program, programming language, header files and | \* |  |  |
|  |  | reserved words; |  |  |  |
|  | 11.1.2 | explain the basic structure of a basic C++ program including |  | \* |  |
|  |  | pre-processor directives, main function, body of program; |  |  |  |
|  | 11.1.3 | define statement terminator and comments in C++ | \* |  |  |
|  |  | programming; |  |  |  |
|  | 11.1.4 | use single-line and multiple-line comments in a C++ program; |  |  | \* |
| 11.2 Constants and Variables in C++ | 11.2.1 | differentiate among C++ data types; |  | \* |  |
| Programming | 11.2.2 | differentiate between variable and constant; | \* |  |
|  | 11.2.3 | write valid variable names on the basis of variables naming |  | \* |
|  |  | rules; |  |  |
|  | 11.2.4 | determine the data types of variables depending upon the |  | \* |
|  |  | values they will store; |  |  |
|  | 11.2.5 | write a C++ program in which variables are declared with |  | \* |
|  |  | different data types; |  |  |
|  | 11.2.6 | write a program to show the implicit and explicit type casting |  | \* |
|  |  | of variables; |  |  |
|  | 11.2.7 | differentiate local and global variables; | \* |  |
| 11.3 Input Output Handling | 11.3.1  11.3.2  11.3.3 | write a program to display message and value of variable using **cout** statement;  write a program for taking input during execution of a program using **cin** statement;  write a program that uses getch(), getche(), gets() and puts() functions for I/ O operations; |  |  | \*  \*  \* |



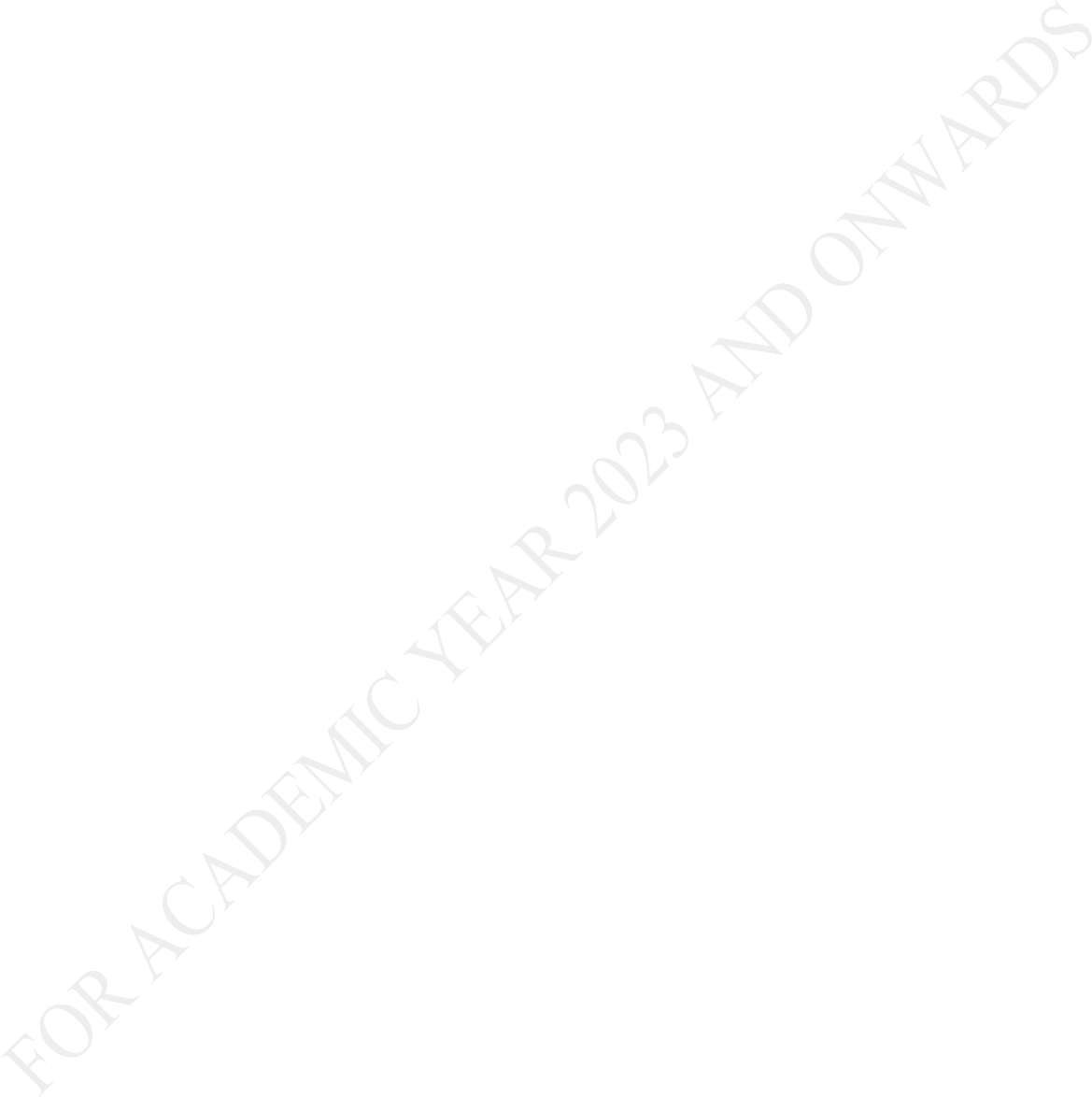
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Topics and Sub-topics** | **Student Learning Outcomes** | | **Cognitive Level** | | |
| **K** | **U** | **A** |
|  | Students should be able to: | | | | |
|  | 11.3.4  11.3.5 | write a program to print text using the following escape sequences:   1. alert – **\a** 2. backspace – **\b** 3. newline – **\n** 4. carriage return – **\r** 5. tab – **\t** 6. display backslash – **\\** 7. display single quotation marks – **\’**;   write a program that uses the **setw** and **endl** manipulators; |  |  | \*  \* |
| 11.4 Operators in C++ Programming | 11.4.1 | differentiate between operator and operand; |  | \* |  |
|  | 11.4.2 | list the names and symbols of arithmetic operators used in | \* |  |  |
|  |  | C++ programming; |  |  |  |
|  | 11.4.3 | write a program using arithmetic operators; |  |  | \* |
|  | 11.4.4 | define assignment operator; | \* |  |  |
|  | 11.4.5 | write a simple program using arithmetic assignment operators; |  |  | \* |
|  | 11.4.6 | write a program using increment and decrement operators with |  |  | \* |
|  |  | the postfix and prefix scenario; |  |  |  |
|  | 11.4.7 | list the names and symbols of relational operators; | \* |  |  |
|  | 11.4.8 | write a program using relational operators; |  |  | \* |
|  | 11.4.9 | write a program using logical operators; |  |  | \* |
|  | 11.4.10 | compare unary, binary and ternary operators; |  | \* |  |
|  | 11.4.11 | write a program to show the order of precedence of arithmetic |  |  | \* |
|  |  | operators in a C++ program; |  |  |  |
|  | 11.4.12 | define compound expression with reference to C++ | \* |  |  |
|  |  | programming; |  |  |  |
|  | 11.4.13 | write a C++ program that uses compound expression. |  |  | \* |



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Topics and Sub-topics** | **Student Learning Outcomes** | | **Cognitive Level** | | |
| **K** | **U** | **A** |
| **12. Control Structures** | Students should be able to: | | | | |
| 12.1 Selection Statements in C++  Programming | 12.1.1 | describe **if**, **if-else** and **else-if** statements; |  | \* |  |
| 12.1.2 | write a program for each **if**, **if-else** and **else-if** statements5; |  | \* |
| 12.1.3 | write a program using **nested if** statement; |  | \* |
| 12.1.4 | explain the use of **switch** statements; | \* |  |
| 12.1.5 | write a program using **switch** statement; |  | \* |
| 12.1.6 | compare **if**, **if-else**, **else-if** and **switch** statement; | \* |  |
| 12.1.7 | describe the role of **default** and **break** keywords; |  | \* |
| 12.1.8 | describe the role of **exit** and **return** functions; | \* |  |
| 12.1.9 | rewrite a program having **if/ if-else/ else-if** statement using  **switch** statement; |  | \* |
| 12.2 Repetition (Loop) in C++ Programming | 12.2.1 | list the types of loops available in C++ programming; | \* | \* |  |
| 12.2.2 | write a C++ program that uses **for** loop; | \* |
| 12.2.3 | write a C++ program that uses **while** loop; | \* |
| 12.2.4 | write a C++ program that uses **do while** loop; | \* |
| 12.2.5 | differentiate between: |  |
|  | a. **for** and **while** loop |  |
|  | b. **while** and **do while** loop; |  |
| 12.2.6 | write a C++ program in which **break** statement is used; | \* |
| 12.2.7 | write a C++ program in which **continue** statement is used; | \* |
| 12.2.8 | write a C++ program in which **exit( )** function is used; | \* |
| 12.2.9 | write a program using **nested for** loop. | \* |

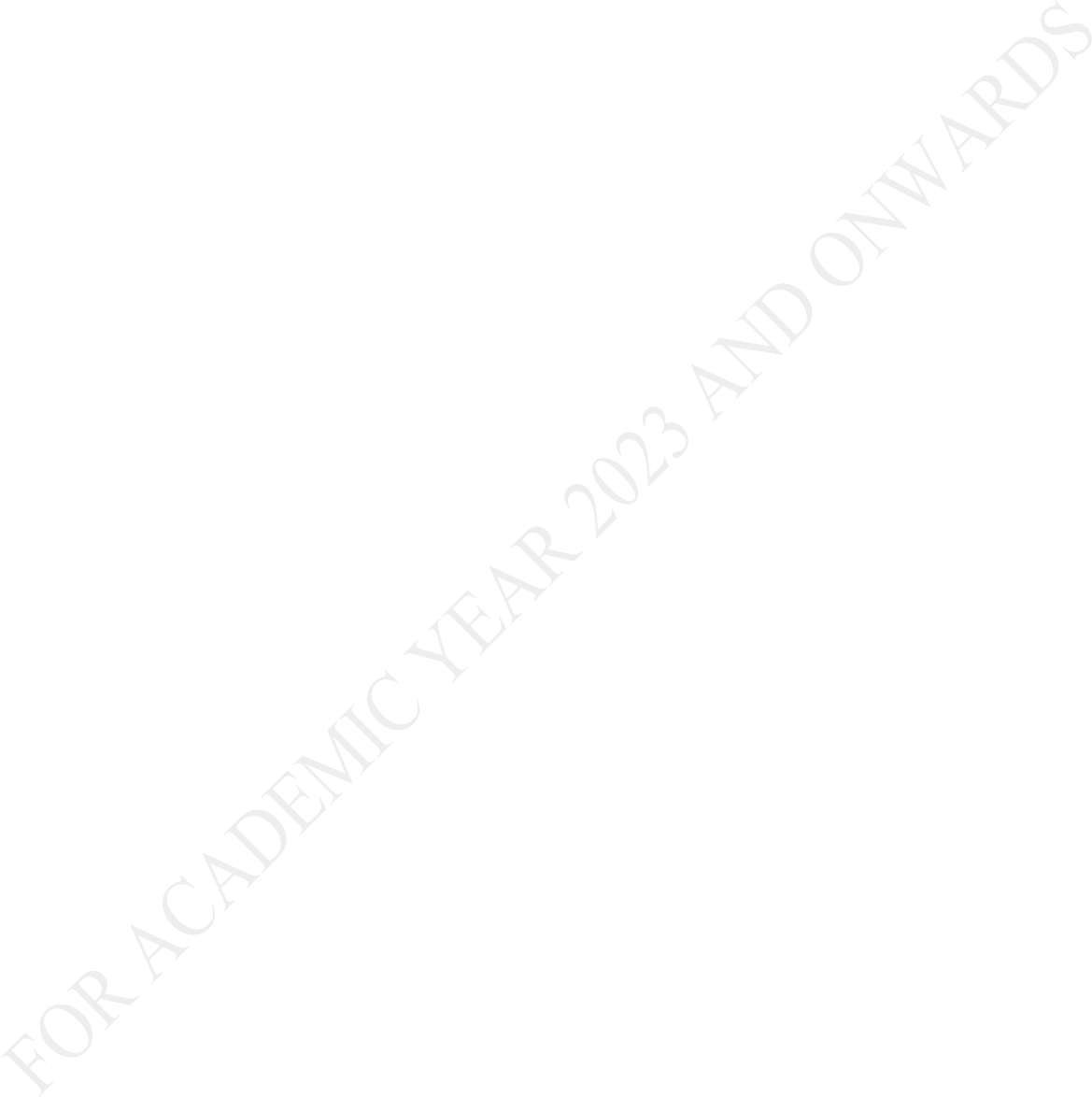


5 Refer to Annex C.

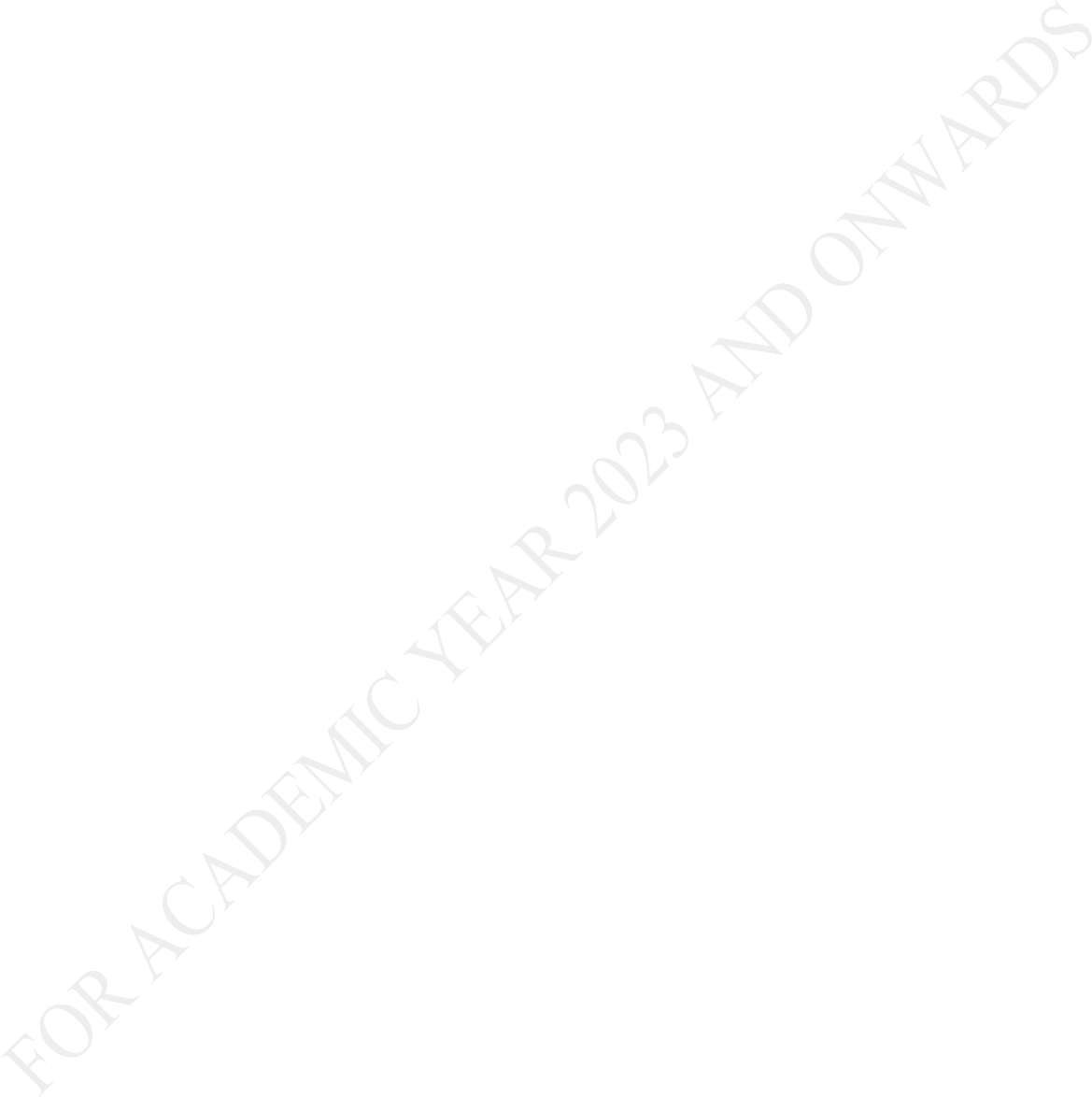


|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Topics and Sub-topics** | **Student Learning Outcomes** | | **Cognitive Level** | | |
| **K** | **U** | **A** |
| **13. Arrays and Strings** | Students should be able to: | | | | |
| 13.1 Introduction to Arrays | 13.1.1 | define array with reference to programming; | \* |  |  |
| 13.1.2 | illustrate the concept of array with respect to array name, size,  index number and arrangement of elements in the memory; | \* |
| 13.1.3 | write C++ code to declare one dimensional array with  different sizes and data types; | \* |
| 13.1.4 | write C++ code to initialise one dimensional array; | \* |
| 13.1.5 | apply process of traversing using all types of loops for input,  output and manipulation of elements; | \* |
| 13.1.6 | use **sizeof( )** function to find the size of an array; | \* |
| 13.1.7 | write a program which stores numeric values in a one dimensional array using **for** loop and finds the highest, lowest and average values; | \* |
| 13.2 Two Dimensional Arrays | 13.2.1 | explain the concept of two dimensional arrays; |  | \* |  |
| 13.2.2 | write C++ code to declare two dimensional array with  different sizes and data types; | \* |
| 13.2.3 | write C++ code to initialise two dimensional array; | \* |
| 13.2.4 | illustrate the process of accessing a particular index number in  two dimensional array and writing value on; | \* |
| 13.2.5 | write a program for adding/ subtracting/ multiplying two integer matrices of the order up to 4×4; | \* |
| 13.3 C++ Strings | 13.3.1 | define string and the process of declaring string variables; | \* | \* | \* |
| 13.3.2 | explain various methods of initializing a string variable; |
| 13.3.3 | write C++ program to perform various operations on string using string functions , i.e. **strcpy**, **strcat**, **strlen** and **strcmp**. |

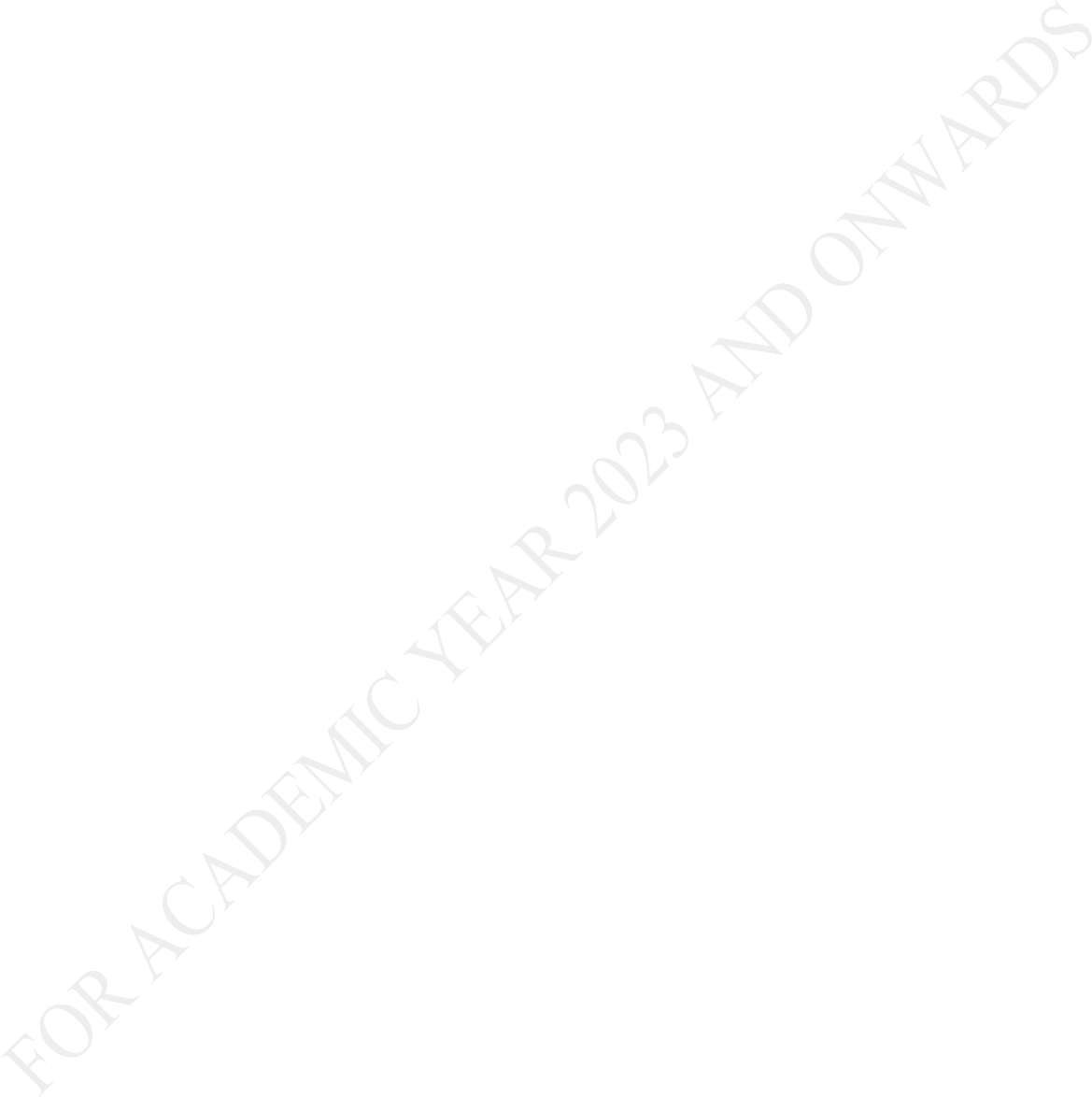
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Topics and Sub-topics** | **Student Learning Outcomes** | | **Cognitive Level** | | |
| **K** | **U** | **A** |
| **14. Functions** | Students should be able to: | | | | |
| 14.1 Introduction to Function in C++ Programming | 14.1.1 | differentiate between predefined and user defined functions; | \* | \* | \* |
| 14.1.2 | describe advantages of user defined functions; | \* |
| 14.1.3 | describe the signature of the functions, i.e. function name,  arguments and returning data type; | \* |
| 14.1.4 | explain the process of creating a user defined function, i.e.  function declaration, function definition and function call; | \* |
| 14.1.5 | write a program involving a user defined function; |  |
| 14.1.6 | differentiate among the variable types, i.e. local, global, and  static; | \* |
| 14.1.7 | differentiate between the formal and actual parameters; | \* |
| 14.1.8 | differentiate between local and global functions; | \* |
| 14.1.9 | define inline function; |  |
| 14.2 Passing Arguments and Returning Values | 14.2.1 | write a program to invoke a user defined function and pass  arguments by constants, value and reference; | \* | \* | \* |
| 14.2.2 | define default argument; |
| 14.2.3 | describe the purpose of return statement; |
| 14.3 Functions Overloading | 14.3.1 | define function overloading; | \* | \* |  |
| 14.3.2 | describe advantages of function overloading. |



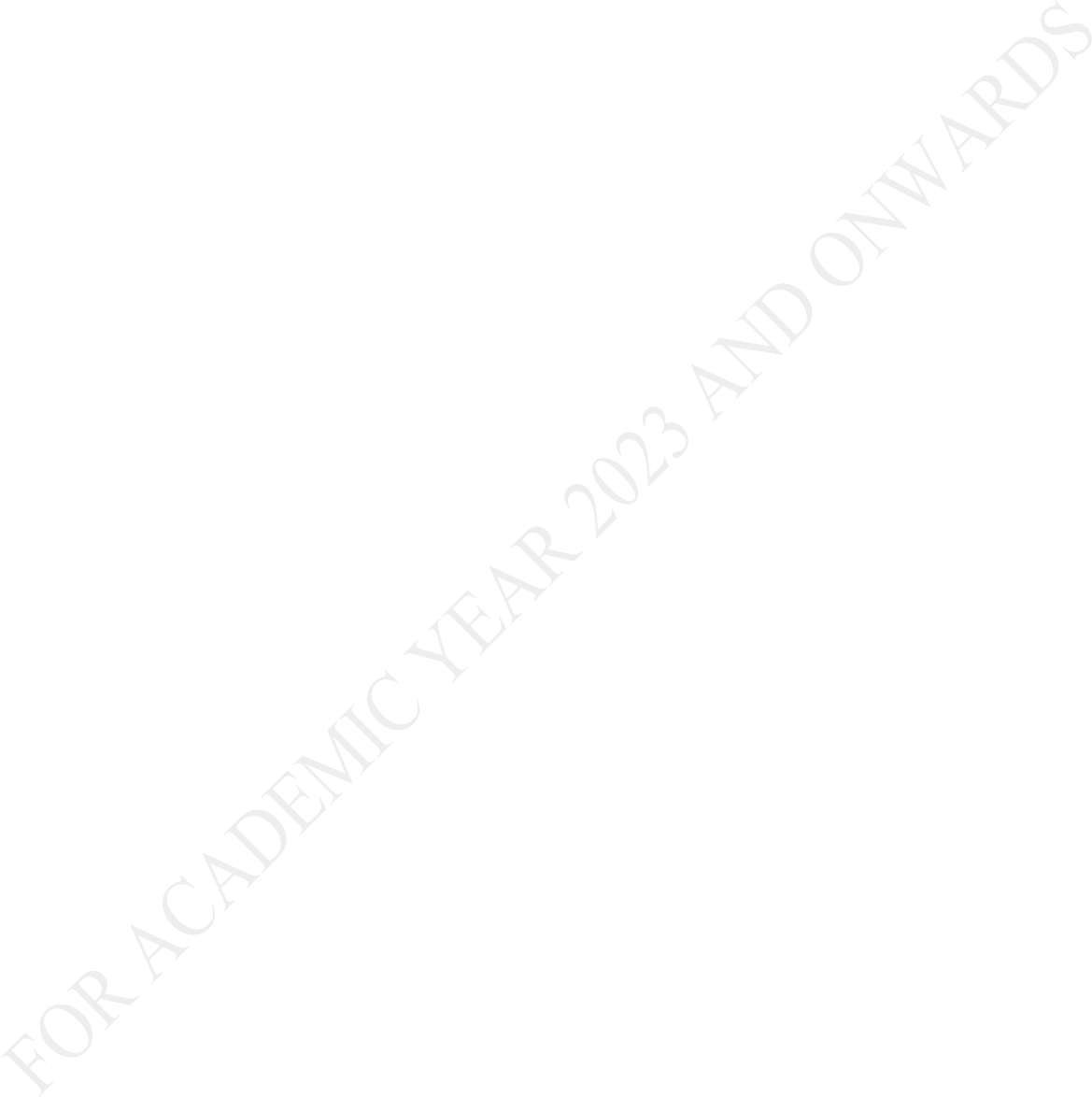
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Topics and Sub-topics** | **Student Learning Outcomes** | | **Cognitive Level** | | |
| **K** | **U** | **A** |
| **15. Pointers** | Students should be able to: | | | | |
| 15.1 Use of Pointers | 15.1.1  15.1.2  15.1.3  15.1.4  15.1.5  15.1.6 | define pointer with respect to C++ programming; describe the purpose of reference operator (&); describe the purpose of dereference operator(\*);  write C++ code to declare an empty pointer variable of **int**, **double**, **float** and **char** data types;  write a simple program using **&** to return memory address of a variable and storing it in a pointer variable;  write a C++ program that uses pointer variable. | \* | \*  \* | \*  \*  \* |



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Topics and Sub-topics** | **Student Learning Outcomes** | | **Cognitive Level** | | |
| **K** | **U** | **A** |
| **16. Object Oriented Programming (OOP)** | Students should be able to: | | | | |
| 16.1 Class**e**s and Objects in OOP | 16.1.1 | explain class and object; |  | \* | \* |
| 16.1.2 | write a C++ program to declare a class along with data members and member functions in its body and create the objects of class in the main( ) function and call member functions of class with the help of objects; |
| 16.2 Access Modifiers (Public, Private, Protected and Sealed) | 16.2.1 | differentiate between private and public access specifiers; |  | \* |  |
| 16.2.2 | write a C++ program in which class members are not  accessible outside the class (private access specifier); | \* |
| 16.2.3 | write a C++ program in which class members are accessible from anywhere where the object is visible (public access  specifier); | \* |
| 16.2.4 | write a C++ program in which a class uses both public and private access specifiers; | \* |
| 16.3 Pillars of OOP (Inheritance, Encapsulation, Abstraction and Polymorphism) | 16.3.1 | explain the concept of encapsulation; | \* | \* |  |
| 16.3.2 | differentiate between constructor and destructor; | \* |  |
| 16.3.3 | differentiate among the types of constructor, i.e. default  constructor, user defined constructor and constructor overloading; | \* |  |
| 16.3.4 | describe inheritance in object oriented programming; | \* |  |
| 16.3.5 | write the basic syntax of inheritance using base class and  derived class; |  | \* |
| 16.3.6 | list the names of three inheritance access specifiers; |  |  |
| 16.3.7 | write the syntax of inheritance for three access specifiers; |  | \* |
| 16.3.8 | illustrate types of inheritance in C++ programming using  different classes; |  | \* |
| 16.3.9 | describe polymorphism in C++ programming; | \* |  |
| 16.3.10 | describe the concept of abstraction in C++ programming; | \* |  |
| 16.3.11 | differentiate between overloading and overriding in OOP. | \* |  |



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Topics and Sub-topics** | **Student Learning Outcomes** | | **Cognitive Level** | | |
| **K** | **U** | **A** |
| **17. File Handling** | Students should be able to: | | | | |
| 17.1 File Handling in C++ Programming | 17.1.1 | differentiate between the binary and the text files in C++  programming; | \* | \* |  |
| 17.1.2 | compare different modes of opening a file; |  |
| 17.1.3 | write programs to show the file opening and closing; | \* |
| 17.1.4 | write a C++ program to read a file character by character and show output; | \* |
| 17.1.5 | write a C++ program to write in a file character by character; | \* |
| 17.1.6 | write a C++ program to read a file into strings. | \* |



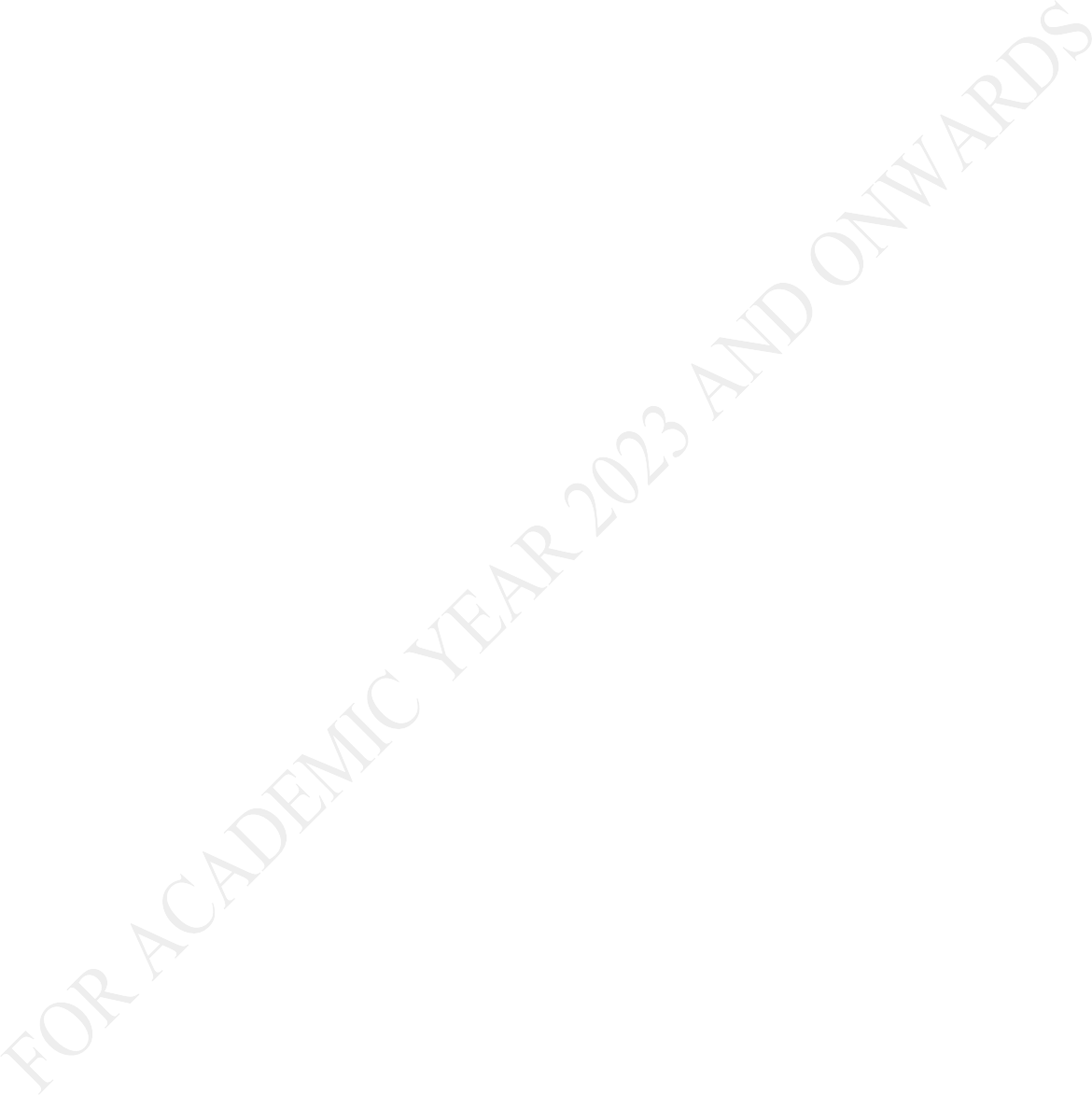
# Grade XII

**Table 3: Number of Student Learning Outcomes by Cognitive Level**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Topic**  **No.** | **Topic** | **No. of Sub-Topics** | **SLOs** | | | **Total** |
| K | U | A | **SLOs** |
| 9. | Operating System (OS) | 3 | 1 | 8 | 1 | 10 |
| 10. | System Development Life Cycle (SDLC) | 1 | 1 | 3 | 0 | 4 |
| 11. | Introduction to C++ Programming | 4 | 6 | 6 | 16 | 28 |
| 12. | Control Structures | 2 | 1 | 6 | 11 | 18 |
| 13. | Arrays and Strings | 3 | 2 | 2 | 11 | 15 |
| 14. | Functions | 3 | 3 | 9 | 2 | 14 |
| 15. | Pointers | 1 | 1 | 2 | 3 | 6 |
| 16. | Object Oriented Programming (OOP) | 3 | 1 | 9 | 7 | 17 |
| 17. | File Handling | 1 | 1 | 1 | 4 | 6 |
|  | **Total** | **21** | **17** | **46** | **55** | **118** |
|  | **Percentage** |  | **14** | **39** | **47** | **100** |

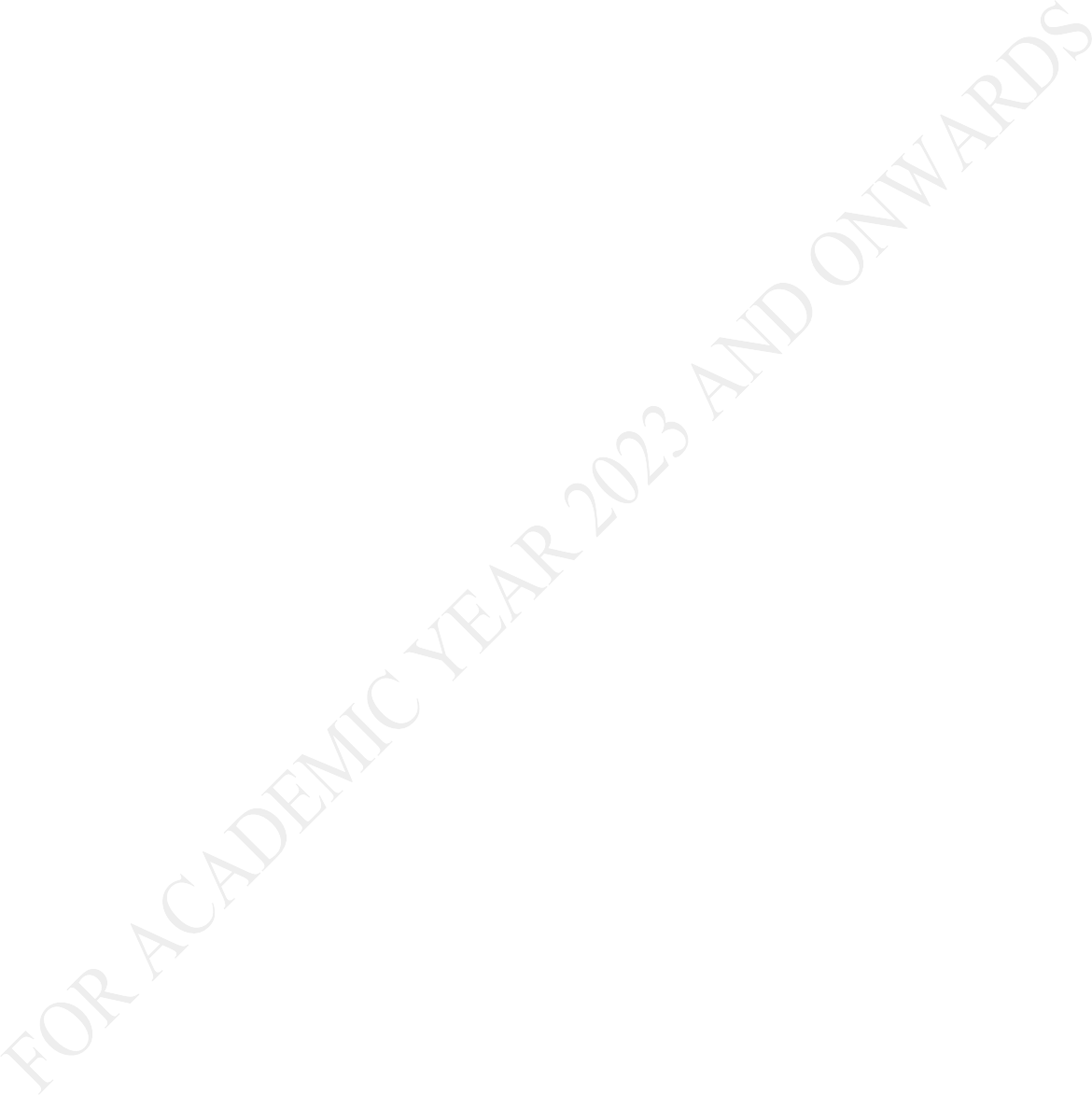
# Table 4: Exam Specification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Topic No.** | **Topics** | **Marks Distribution** | | | **Total Marks** |
|  |  | **MCQs** | **CRQs** | **ERQs** |
| 9. | Operating System (OS) | 6 | Total 3 Marks  (1 CRQ) |  | 9 |
| 10. | System Development Life Cycle (SDLC) | 2 | Total 3 Marks  (1 CRQ) |  | 5 |
| 11. | Introduction to C++ Programming | 10 | Total 5 Marks  (1 CRQ) | 7 Marks Choose any ONE from TWO | 33 |
| 12. | Control Structures | 8 | Total 3 Marks  (1 CRQ) |
| 13. | Arrays and Strings | 3 |  | 7 Marks Choose any ONE from TWO | 17 |
| 14. | Functions | 7 |  |
| 15. | Pointers | 3 |  |  | 3 |
| 16. | Object Oriented Programming (OOP) | 8 | Total 4 Marks  (1 CRQ) |  | 12 |
| 17. | File Handling | 3 | Total 3 Marks  (1 CRQ) |  | 6 |
|  | **Total** | **50** | **21** | **14** | **85** |
|  | **Practical\*** |  |  |  | **15** |
|  | **Total** |  |  |  | **100** |

* Multiple Choice Question (MCQ) requires candidates to choose one best/ correct answer from four options for each question. Each MCQ carries ONE mark.
* Constructed Response Question (CRQ) requires students to respond with a short text (few phrases/ sentences), calculations or diagrams.
* Extended Response Question (ERQ) requires students to answer in a more descriptive form. The answer should be in paragraph form, with diagrams where needed, and address all parts of the question.

**Annex B: Examples of Entity-Relationship Diagram (ERD)6**

|  |
| --- |
| **Entity Relationship Diagram** |
| 1. Library Management System 2. Student Management System 3. Hotel Management System 4. Hospital Management System 5. Ticket Booking System |

**Annex C: Examples of C++ Programming7**

|  |
| --- |
| **Selection Statement** |
| 1. Generate marksheet of students on the basis of inputted marks of different subjects. 2. Show whether a number is positive, negative or zero. 3. Find the maximum and minimum values from inputted numbers. 4. Show whether a number is even or odd. 5. Generate the utility bill on the basis of charges allocated to each unit range. The bill should contain meter number and name of the consumer 6. Identify whether the inputted string is a palindrome or not |
| **Loops** |
| 1. Generate a number series (even, odd, prime, Fibonacci etc.) by taking input of starting and ending point. 2. Generate sum of series using loops. 3. Generate table of any inputted number. 4. Calculate factorial of any inputted number. 5. Print pyramid, rectangle and square or any other geometrical shape using nested loops. 6. Input multiple values using loop and calculate average, maximum or minimum value using selection statement. 7. Calculate the number of characters, vowels in an inputted string 8. Reverse an inputted string |

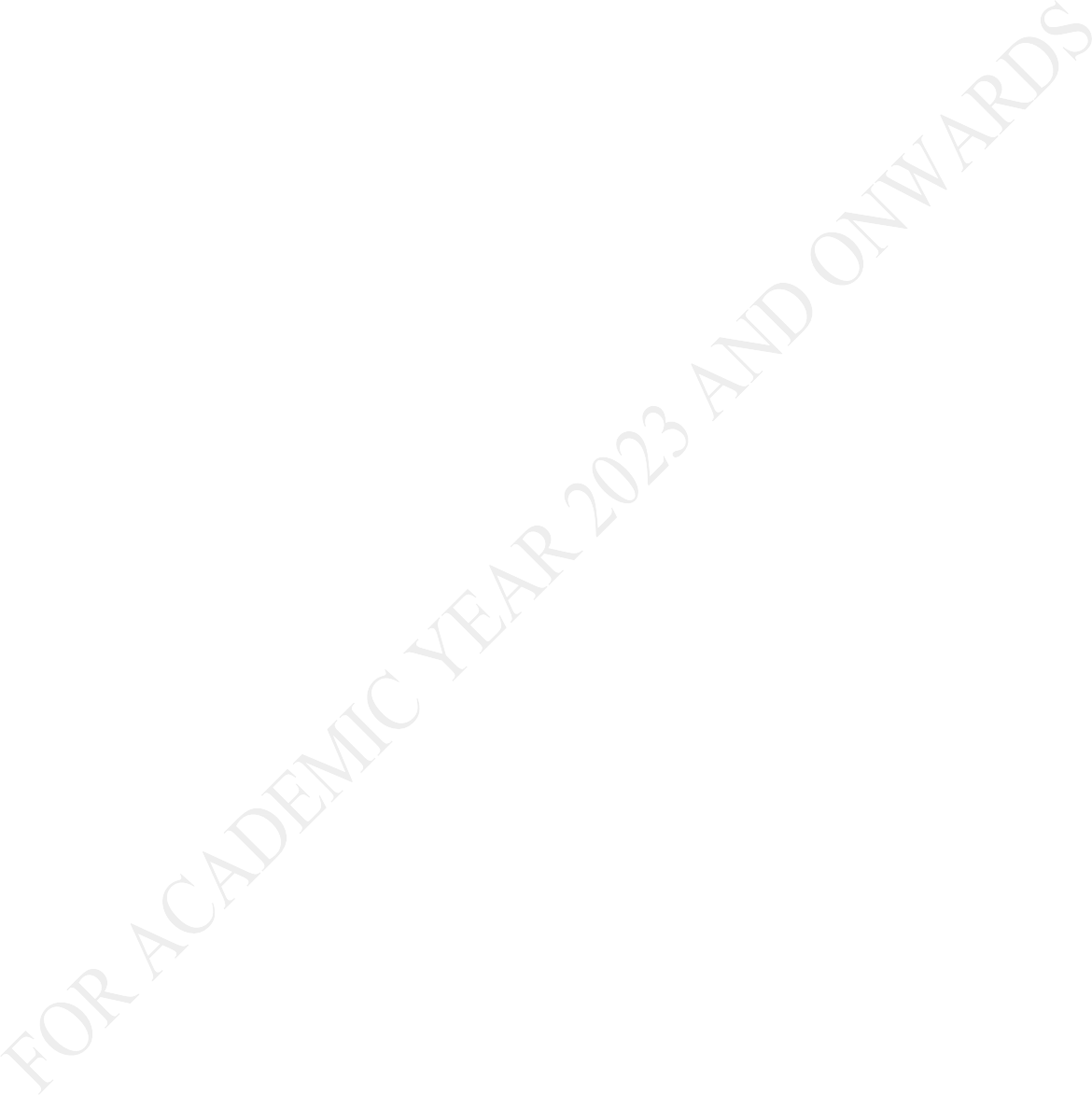
6Government of Pakistan (2009), Page 34, *National Curriculum for Computer Science IX-XII, Islamabad,*

Ministry of Education (Curriculum Wing)

7 Government of Pakistan (2009), Page 48, *National Curriculum for Computer Science IX-XII, Islamabad,*

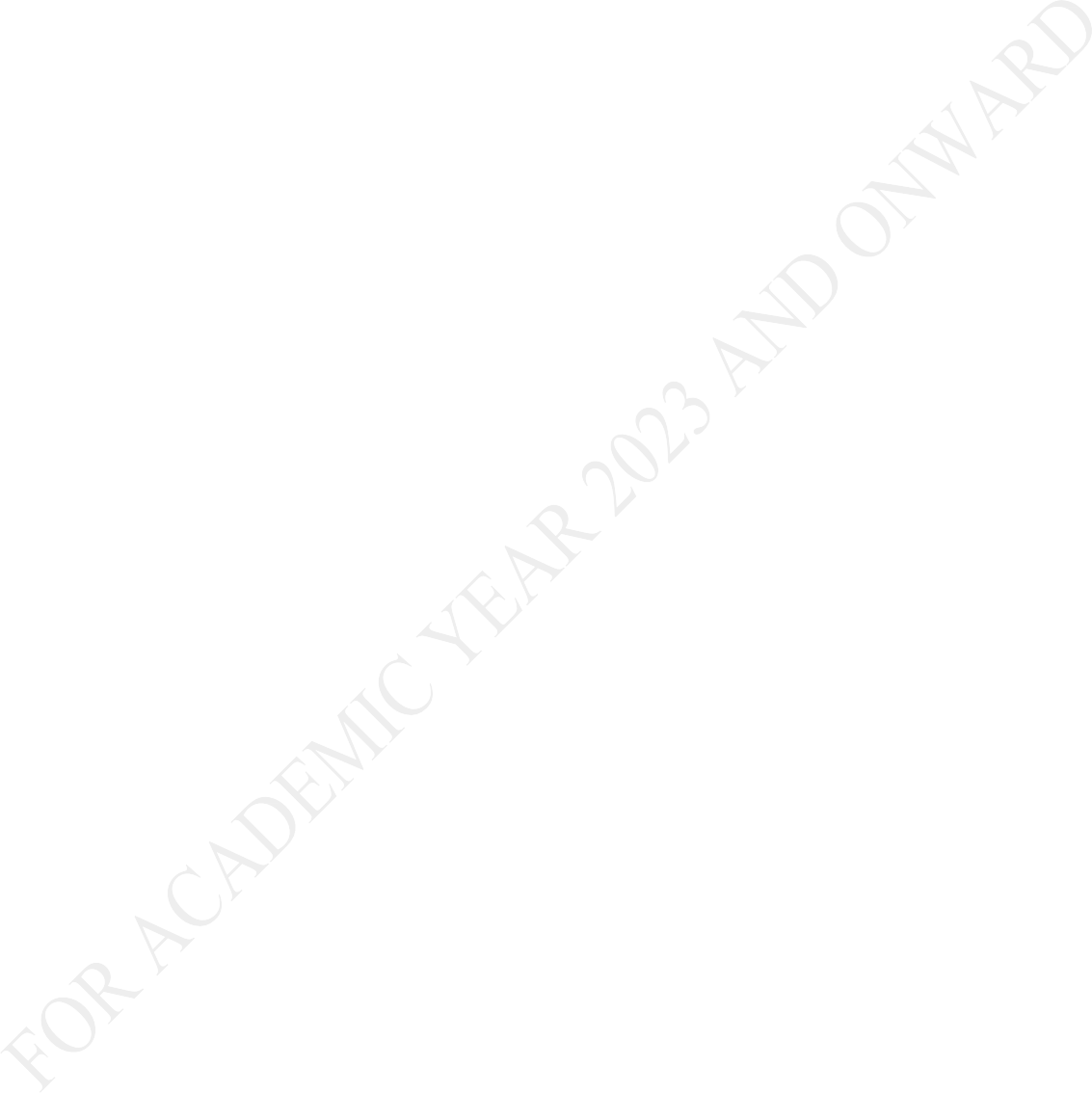
Ministry of Education (Curriculum Wing)

|  |
| --- |
| **Functions and Classes** |
| 1. Write a user defined function to perform basic arithmetic operations, i.e. add, subtract, multiply and divide. 2. Write a user defined function to calculate area of circle, triangle, parallelogram or any other geometrical shape. 3. Write a user defined function to calculate area, volume of cylinder, sphere, cube and different geometrical shapes. 4. Write a user defined function to calculate average of numbers. 5. Write a user defined function to calculate factorial of a given number. 6. Write C++ programs to define the classes and objects and call them. 7. Write C++ programs to show the concept of inheritance |



# Class XII

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No** | **SLO No** | **Objective** | **Equipment** | **Software** |
|  |  | **Topic 12: Control Structures** | | |
| 1. | 12.1.2 | Write a program for each **if**, **if-else** and **else-if**. | Computer | Dev-C ++ or Any Other C Compiler |
| 2. | 12.1.3 | Write a program using **nested if** statement. |
| 3. | 12.1.5 | Write a program using **switch** statement. |
| 4. | 12.2.2 | Write a C++ program that uses **for** loop. |
| 5. | 12.2.3 | Write a C++ program that uses **while** loop. |
| 6. | 12.2.4 | Write a C++ program that uses **do while** loop. |
|  |  | **Topic 13: Arrays and Strings** | | |
| 7. | 13.1.7 | Write a C++ program which stores numeric values in a one dimensional array using  **for** loop and finds the highest, lowest and average values. | Computer | Dev-C ++ or Any Other C Compiler |
| 8. | 13.2.5 | Write a C++ program for adding/ subtracting/ multiplying two integer matrices of the order up to 4x4. |
| 9. | 13.3.3 | Write C++ program to perform various operations on string using string fucntions ,  i.e. **strcpy**, **strcat**, **strlen** and **strcmp**. |



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No** | **SLO No** | **Objective** | **Equipment** | **Software** |
|  |  | **Topic 14: Functions** | | |
| 10. | 14.1.5 | Write a program involving user defined function to perform basic arithmetic operations, i.e. add, subtract, multiply and divide. | Computer | Dev-C ++ or Any Other C Compiler |
| 11. | 14.1.6 | Write a program involving user defined function to calculate area of circle, triangle and parallelogram. |
| 12. | 14.1.7 | Write a program involving use of user defined function to calculate volume of cylinder, sphere and cube. |
| 13. | 14.1.9 | Write a program involving user defined function to calculate factorial of a given number. |
| 14. | 14.1.8 | Write a program involving user defined function to calculate average of numbers. |
|  |  | **Topic 15: Pointers** | | |
| 15. | 15.1.5 | Write a simple program using **&** to return memory address of a variable and storing it in a pointer variable. | Computer | Dev-C ++ or Any Other C Compiler |
| 16. | 15.1.6 | Write a C++ program that uses pointer variable. |
|  |  | **Topic 16: Object Oriented Programming (OOP)** | | |
| 17. | 16.1.2 | Write a C++ program to declare a class along with data members and member functions in its body and create the objects of class in the main( ) function and call  member functions of class with the help of objects. | Computer | Dev-C ++ or Any Other C Compiler |
| 18. | 16.1.6 | Write a C++ program in which a class uses both public and private access specifiers. |