Federal Board **HSSC-II** Examination
Computer Science Model Question Paper (2nd Set Solution)
(Curriculum 2009)

SOLUTION SECTION – A (Q:1)

Q.1 Choose the correct answer i.e. A / B / C / D by filling the relevant bubble for each question on the OMR Answer Sheet according to the instructions given there. Each part carries one mark.
(Answers are underlined)

1. Which of the following phase of SDLC involves training of personnel to use the new system?
   A. System Analysis ○ B. **System Implementation** ●
   C. System Design ○ D. System Coding ○

2. Assume **value** is an integer variable. If the user enters **333.14** in response to the following programming statement, what will be stored in value? **cin >> value;**
   A. 333 ● B. 33314 ○
   C. 333.14 ○ D. Compile time error ○

3. Which of the following Operating system is time bound and has a fixed deadline?
   A. Embedded OS ○ B. Distributed OS ○
   C. Time sharing OS ○ D. **Real time OS** ●

4. What will be the last phase in SDLC?
   A. Planning ○ B. Design ○
   C. **Maintenance** ● D. Coding ○

5. Which of the following header file is included in program to use **setw( )** function?
   A. iostream.h ○ B. stdio.h ○
   C. **iomanip.h** ● D. conio.h ○

6. What will be the output of following program segment?
```
        for (i = 1 ; i < 10 ;  i = i + 5) ;
            cout << i << " \ t ";
```
   A. **11** ● B. 1 6 ○
   C. 1 6 11 ○ D. Infinite iterations ○

7. In the following array definition, what value is stored in **number [4]**?
```
        int number [5] = { 1, 2, 3 };
```
   A. **0** ● B. 1 ○
   C. 2 ○ D. 3 ○

8. What will be printed after executing the following code?
```
    int x = 5;
    if ( x++ = = 5)
        cout << " Five ";
    else
            if ( ++x = = 6)
                cout << " Six ";
```
   A. **Five** ● B. Six ○
   C. FiveSix ○ D. Compile time error ○

9. Which of the following is a valid declaration, if a string needs to be store **10** characters?

A. char S [0] ○ B. char S [10] ○
[✓ C. **char S [11]**] ● D. char S [12] ○

10. How many values a function can return at a time?

[✓ A. **One value**] ● B. Two values ○
C. Three values ○ D. Any number of values ○

11. A pointer is a(n):

A. Operator ○ B. Data type ○
C. Keyword ○ [✓ D. **Variable**] ●

12. Which of the following pointer hold address of any type and can be type-casted to any data type:

A. data ○ [✓ B. **void**] ●
C. NULL ○ D. function ○

13. Which of the following is **FALSE** about destructor?

A. It does not accept any argument ○ B. It deallocates memory of an object ○
[✓ C. **It can be overloaded**] ● D. There is always one destructor ○

14. Ofstream is a(n):

A. Command ○ [✓ B. **Class**] ●
C. Object ○ D. Method ○

15. Which of the following is used to open a file for writing and move the read/write control immediately to the end?

A. ios :: ate ○ B. ios :: in ○
[✓ C. **ios :: app**] ● D. ios :: out ○

SOLUTION SECTION – B (Q:2)

Part – i: Write down three differences between multi-threading and multi-tasking.

Answer: (any three valid points)

	Multi-threading	**Multi-tasking**
1.	In multithreading, CPU switching is involved between the threads	Multitasking involves CPU switching between the tasks
2.	Threads share same memory and resources	Processes share separate memory
3.	Multithreading is faster	Multitasking is slow as compared to multithreading
4	Termination of thread takes less time.	Termination of process takes more time.

Part – ii: What is the role of Testing/ verification phase in System Development Life Cycle?

Answer: **ROLE OF TESTING/ VERIFICATION PHASE** (at least any three valid points)

- Testing/ verification phase in SDLC examines the system for functionality and quality by discovering problems and errors.
- It evaluates the capabilities of code/programs.
- It helps to verify proper integration and interaction of each component in the system.
- It helps in reviewing of requirements, design, and code.
- It helps in identifying defects and ensuring they are addressed before system deployment.
- It also plays important role in lowering the maintenance cost of the system.

Part – iii: What is a named constant?
Write statements for the following values that create named constants: a) 3.14159 b) 1609

Answer: Named constant

A literal/ constant may be given name to represent in a program. A named constant is like a variable, but its content is read-only and cannot be changed while the program is running. It is defined with a qualifier **const.**

a) 3.14159

```
const float PI = 3.14159;
```

b) 1609

```
const int A = 1609;
```

Part – iv: The following C++ code has compile time errors. The line numbers are written along the left column. They are not part of the program code. Identify and correct errors:

```
1    \* identify and correct errors *\
2    void main(void)
3    {
4        int a, b, s
5        char c = A;
6        cout << ”enter two numbers;
7        cin >> a >> b;
8        a + b = s;
9        cout << \n << s ;
10   }
```

Answer: Correct code (correction of six errors, each of 0.5 mark)

Line no.	Error identification	Error free Code
1	Wrong comment symbol	**/*** identify and correct errors ***/**
2		void main(void)
3		{
4	Missing statement terminator **(;)**	int a, b, s **;**
5	Wrong character variable initialization	char c = **’A’**;
6	Unterminated string constant **(”)**	cout << ”enter two numbers ”;
7		cin >> a >> b;
8	Wrong expression syntax	**s = a + b ;**
9	Wrong use of escape sequence	cout << **“\n”** << s ;
10		}

Part – v: Differentiate between declaration and initialization of a variable with example.

Answer:

	Declaration of a variable	**Initialization of a variable**
1.	Variable declaration specifies the name and data type of a variable to a compiler	Variable is initialized when it is assigned a value before use
2.	Variable declaration gives details about the properties of a variable	The specified initial value stored in a variable to avoid garbage data

3.	Example:
	int s;
	char ch;
		Example:
		int s = 0; // initialize to zero
		char ch = 'Y'; // initialize to 'Y'

Part – vi:
a) Write an if-else statement that assigns **1** to **x** if **y** is equal to **100**, otherwise it should assign **0** to **x**.

Answer:

```
if ( y = = 100)
      x = 1;
else
      x  = 0;
```

b) The following statement should determine if count is outside the range of 0 through 100. What is wrong with it?   if (count < 0 && count > 100)

Answer:

```
if (count < 0 || count > 100)
```

Logical OR **(||)** operator is required to determine, if count is outside the range of 0 through 100.

Part – vii: Write down three differences between break statement and exit( ) function.

Answer: (any three valid points)

	break statement	**exit( ) function**
1.	It is a keyword and does not require any header file to include	It is a built-in function requires a header file stdlib.h to include
2.	It is often used to terminate from the control structure like loop, switch statement	It is often used to terminate the program from anywhere in a program
3.	More than one break statements can be executed in a program	Only one exit( ) function will be executed in a program
4.	**Syntax**: break; // exits from control structure	**Syntax**: exit (0) ; // exits program without any error

Part – viii: Write down the output of the following program segments.

a)	b)

```
int funny = 7, serious = 15;
funny = serious % 2;
switch (funny)
{
 case   1:    cout << "That is funny";
                    break;
 case   7:    cout << "That is serious";
                    break;
 case   30:  cout << "That is seriously funny";
                    break;
 default :    cout << funny << endl;
}
```
	```
	int i , j;
	for(i = 1; i <= 4; i++)
	{
	     for(j = 1; j <= i; j++)
	              cout << i * i  << "   ";
	     cout << "\n" ;
	}
	```

Answer:

a) **That is funny**

b)
```
1
1    4
1    4    9
1    4    9    16
```

Part – ix: What is the purpose of sizeof ( ) function? How do you find the size of an array?

Answer:

The sizeof ( ) function returns the size of the passed variable in memory. It provides the number of bytes occupied by a variable according to the data type. It returns 2 for integer, 4 for float etc.

Size of an array:

It also returns the total number of bytes occupied by each element of an array.

Example:

```
int   array [10] ;
cout << "size of an array is :" << sizeof (array);
```

Output: Size of an array is: **20**

Note: Number of bytes reserved may vary depending on compiler (like turbo C++ reserves 2 bytes for each integer element and Dev C++ reserves 4 bytes, both answers will be considered correct)

Part – x: Write down any three differences between string and array.

Answer: (any three valid points)

	String	Array
1.	A string can hold items of only the character data type, hence considered as a set of characters.	Arrays can hold items of any numeric data type such as set of integers, set of floating-point numbers.

2.	String terminates with a null character ('\0').	Arrays does not terminate with a null character.
3.	String size can have variable number of elements.	Arrays are of fixed size.
4.	Syntax: char name [20];	Syntax: float marks [10];

Part – xi: What will be display after executing the following code?

```
int test (int x, int y)
{
        return  y % x ;
}
void main(void)
{
        int a = 3, b = 70, c , d ;
        c = test(a , b);
        d = test(b , a);
        cout << b << "\t" << c << "\t" << d << endl;
}
```

Answer: (one mark for each value)

```
70    1    3
```

Part – xii: Compare local and static variables in terms of scope, lifetime, and storage duration.

Answer:

	Local variable	**Static variable**
1.	**Scope:** A local variable has block/ local scope, only limited to the function where it is defined.	A static local variable exists only inside a function where it is declared (like a local variable).
2.	**Lifetime:** It is created at the point of definition and destroyed at the end of the block in which it is defined.	Its lifetime starts when the function is called and ends only when the program ends
3.	**Storage duration:** It has automatic storage duration determines its lifetime and destroyed when the function exits.	Its space is allocated only once in function and the value of variable gets carried from previous function call to the next function call

Part – xiii: Write down the purpose of asterisk (*) in the following statements:

Answer:

a) distance = speed ***** time;

   * used as the **multiplication** arithmetic operator. It gives the product of speed and time.

b) int *****ptr = &n;

   * used as the **definition/ declaration of a pointer** variable. It assigns address of variable **n** to a pointer variable **ptr.** (Affecting variable **ptr** by storing memory address)

c) *****ptr = 100;

   * used as the **dereference** operator. It assigns **100** to the memory location whose address is stored in **ptr** variable (Not affecting ptr, but the value **ptr points to**).

Part – xiv: What is the relationship between a class and an object?

Answer:

Class:
A class is a template for defining objects. A class is a user-defined datatype that has its own data members and member functions. It defines the structure, properties, behavior, and contents of the objects that belong to it. It specifies how instances are created and how they behave.

Object:
An object is an instance of a class by which we can access the data members and member functions of the class. An object exhibits the properties and behaviors defined by its class. Multiple objects can be created from one class.

Example: Car is a class with properties and functions. Honda and Toyota are objects of Car.

Part – xv: Write down three differences between a text file and a binary file?

Answer: : (any three valid points)

	Text file	**Binary file**
1.	In text file, text, character, numbers are stored one character per byte in memory.	In binary file data is stored in binary format in memory.
2.	Text files are used to store data more user friendly.	Binary files are used to store data more compactly.
3.	In text file, a special character inserted after the last character to mark the end of file.	In binary file no such character is present.
4.	Content written in text files is human readable.	Content written in binary files is not human readable and looks like encrypted content.

Part – xvi: What is stream? Write down the purpose of the following functions:
a) getline( ) b) get( )

Answer:

Stream:

C++ Input Output are based on streams, which are sequence of bytes flowing in and out of the programs. In input stream, data bytes flow from an input source (e.g., keyboard, file) into the program. In output stream, data bytes flow from the program to an output sink (e.g. console, file).

a) getline( )

getline( ) is a standard library function defined in <fstream.h> used to read a string or a line from an input stream. It extracts characters from the input stream and appends it to the string object. For example:

file1.getline(name,50); // reads a string **name** of 50 characters from file linked to **file1**

b) get( )

get( ) is a standard library function defined in <fstream.h> used to handle a single character. It is used to fetch or read a character from file including blank space, tab, new line character etc.

For example: file2.get(ch); // reads a character **ch** from file linked to **file2**

SOLUTION SECTION – C

Note: Attempt any **THREE** questions. All questions carry equal marks. (3 × 8 = 24)

Q.3 (i) Explain the following functions of Operating System:
a. Memory management b. File Management

Answer:

a. Memory management

An operating system manages programs in the main memory and disk during process execution. Memory is used to store instructions and processed data. The CPU fetches instructions from the memory. Memory management is required to achieve multiprogramming and proper utilization of memory. The operating system resides in a part of memory and the rest is used by multiple processes. The following are some of the tasks performed by operating system:

- Allocate and de-allocate memory before and after process execution.
- Keep track of used memory space by processes.
- Minimize fragmentation issues.
- Proper utilize main memory by swap-in and swap-out processes from memory.

b. File management

An operating system is used to manage files and folders of computer system. A file is a collection of specific information stored on secondary storage. File management is the process of manipulating, creating, modifying, and deleting the files in computer system. The following are some of the tasks performed by operating system:

- Create new files in computer system and place them at specific locations.
- Help to share files among different users.
- Store and manage the files in separate folders.

(ii) Write down four differences between pretest and posttest loops.

Answer: (any four valid points)

	Pre-test loop	**Post-test loop**
1.	The test condition is evaluated before executing any statement within the loop.	The test condition is evaluated after executing all statements within the loop.
2.	If the test condition is false, the loop statements/ body of loop never execute.	Even, if the test condition is false, the loop statements/ body of loop execute once.
3.	It is considered as entry-controlled structure.	It is considered as exit-controlled structure.
4.	The **for** and **while** loop are pre-test loops.	The **do-while** loop is post-test loop.

5.	Example:
	cin >> marks;
	while (marks > 0)
	{
	    s = s + marks;
	    cin>>marks;
	}
		Example:
		cin >> marks;
		do
		{
		    s = s + marks;
		    cin>>marks;
		}
		while (marks > 0);

Q.4 (i) Write down four responsibilities of any two personnel involved in SDLC.

Answer: (any four valid points of each)

1. **System Analyst Responsibilities:**

   i. Defines technical requirements by consulting with clients, evaluating procedures and processes
   ii. Implements computer system requirements by defining and analyzing system problems
   iii. Interacts with designers to understand software limitations
   iv. Develops solution by preparing and evaluating alternative workflow solutions
   v. Provides reference by writing documentation

2. **Project Manager Responsibilities:**

   i. Ensures the development team has everything they need to get the work done.
   ii. Manages meetings and communication with the client and the software development team.
   iii. Distributes the tasks among the members of development team
   iv. Controls the development process, and coordinates team activities.
   v. Responsible for time management and risk management etc.

(ii) Explain two-dimensional array. Initialize two-dimensional arrays of two different data types.

Answer:

Two-dimensional array:

A two-dimensional (2D) is an array of arrays. It is useful for storing multiple sets of data. It is like several identical arrays put together. It is also known as matrix or table format, having rows and columns of elements.

	Column[0]	Column[1]	Column[2]	Column[3]
Row[0]	X[0][0]	X[0][1]	X[0][2]	X[0][3]
Row[1]	X[1][0]	X[1][1]	X[1][2]	X[1][3]
Row[2]	X[2][0]	X[2][1]	X[2][2]	X[2][3]

The array has three rows (numbered 0 through 2), and four columns (numbered 0 through 3). There are a total of 12 elements in the array. Following is the declaration of two-dimensional array with two size declarators, one is for number of rows, and other is for the number of columns.

float X [3][4];

float represents, elements of floating type data

X is the name of an array

[3] represents number of rows

[4] represents number of columns

Initialization of Two-dimensional array:

i.
```
int scores[3][4] = {   {10, 20, 30, 40},
                       {14, 32, 52, 16},
                       {50, 24, 31, 47}  };
```

ii.
```
float numbers[2][3] = {  {71.5, 28.2, 30.6 },
                         {9.34, 6.32, 85.6}  };
```

Q.5 Write a program that computes and displays the charges for a patient's hospital stay. The program should ask if the patient was admitted as an in-patient or an out-patient.

- If the patient was an in-patient, the following data should be entered:
  - The number of days spent in the hospital
  - The daily rate
  - Hospital medication charges
  - Charges for hospital services (lab tests, etc.)
- If the patient was an out-patient the following data should be entered:
  - Charges for hospital services (lab tests, etc.)
  - Hospital medication charges

The program should use **two overloaded functions** to calculate the total charges. One of the functions should accept arguments for the in-patient data, while the other function accepts arguments for out-patient information. Both functions should return the total charges to the main function.

Answer:
```
#include <iostream.h>
#include <conio.h>
float patient(int days, float rate, float medcharges, float hscharges);
float patient(float medcharges, float hscharges);
void main(void)
{
 char p;
 int days;
 float rate, medcharges, hscharges, total;
 cout<<"Enter I for in-patient or O for out-patient: ";
 cin>>p;
 if(p == 'I' || p == 'i')
 {
     cout<<"enter Number of days spent in the hospital: ";
     cin>>days;
     cout<<"enter Daily rate: ";
     cin>>rate;
     cout<<"enter Hospital medication charges: ";
     cin>>medcharges;
     cout<<"Charges for hospital services: ";
     cin>>hscharges;
     total = patient(days, rate, medcharges, hscharges);
     cout<<"Patient total charges: "<<total;
   }
else
      if(ans == 'O' || ans == 'o')
         {
            cout<<"enter Hospital medication charges: ";
            cin>>medcharges;
            cout<<endl;
            cout<<"enter Charges for hospital services: ";
            cin>>hscharges;
            total = patient(medcharges, hscharges);
            cout<<"Patient total charges: "<<total;
         }
   getche();
 }

 float patient(int days, float rate, float medcharges, float hscharges)
 {
    float totalcharges;
    totalcharges = (days  * rate) + medcharges + hscharges;
    return totalcharges;
 }

 float patient(float medcharges, float hscharges)
 {
    float totalcharges;
    totalcharges = medcharges + HScharges;
    return totalcharges;
 }
```

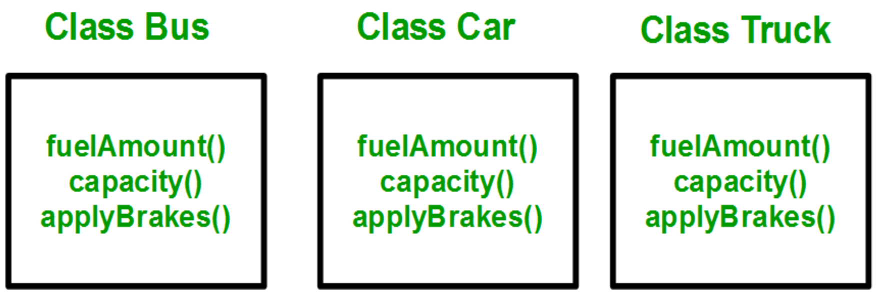
Q.6 (i) Explain inheritance with daily life example.

Answer:

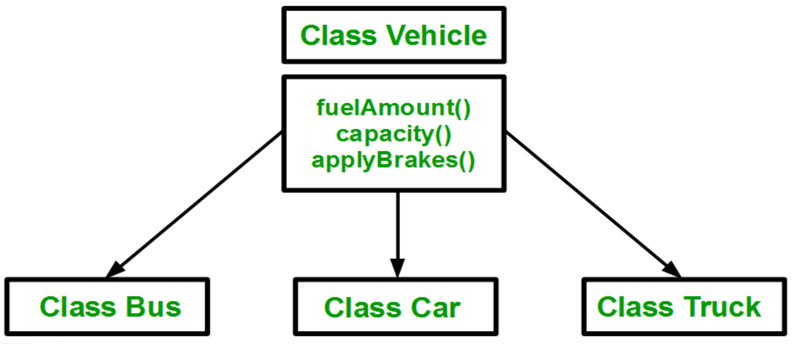
Inheritance:

Inheritance is one of the most important features of Object-Oriented Programming. The capability of a class to derive properties and characteristics from another class is called Inheritance. Inheritance allows a new class to be based on an existing class. The new class inherits all the member variables and functions (except the constructors and destructor) of the class. The class that inherits properties from another class is called **Sub class** or **Derived Class**. The class whose properties are inherited by sub class is called **Base Class** or **Super class**.

Consider a group of vehicles. We need to create classes for **Bus**, **Car** and **Truck**. The functions fuelAmount( ), capacity( ), applyBrakes( ) will be same for all of the three classes. If we create these classes avoiding inheritance, then we must write all these functions in each of the three classes.



This results in duplication of same code, increases data redundancy. To avoid this type of situation, inheritance is used. We create a class **Vehicle** and write these three functions in it and inherit remaining classes from the vehicle class. It will avoid the duplication of data and increase re-usability. Three classes are inherited from **vehicle** class:



(ii) Define a class declaration named **Inventory** in a retail store with following members:

- Private members named item number, quantity, price, and total cost.
- Constructor to the class that initialize item number, quantity, and price to **0**.
- Public member function **get** to accept data of item number, quantity, and price.
- Public member function **display** to calculate and print total cost of inventory

Answer:

```
 // Inventory class declaration
class inventory
{
  private:                    // declarations of private members
        int       item_number;
        int       quantity;
        float     price;
        float     total_cost;

  public:                    // declarations of public members
       inventory ( )
       {
           item_number = 0;
           quantity = 0;
           price = 0;
       }
       void get (void)
       {
            cout<< "enter item number, quantity and price:"
            cin >> item_number;
            cin>> quantity;
            cin>> price;
       }
       void display (void)
       {
            total_cost = quantity * price;
            cout<< "total cost of inventory:" << total_cost;
        }
  };
```

<u>NOTE</u>:

This is suggested (proposed) solution to the questions given in <u>SECTION-B and C</u>. Students can write any valid alternate answers.