INDIAN SCHOOL MUSCAT

HALF YEARLY EXAMINATION

SET A

SEPTEMBER 2019

CLASS XII

Marking Scheme – COMPUTER SCIENCE [THEORY]

Q.NO.	Answers	Marks (with split up)
1(a)	(i) iostream.h (ii) string.h (½ Mark each for writing correct header file)	1

(b)	#define Area(L,B) L*B //Error 1
	 struct Recta //Error 2
	{ int Length, Breadth; };
	void main()
	{ Recta R = {10,15}; //Error 3
	  cout<<Area(R.Length,R.Breadth); //Error 4
	
	(½ Mark for correcting each Error and rewriting the statement correctly)
		2

(c)	While, Float, Amount2, _Counter (½ Mark for each correct identifier)	2
(d)	(iii). 100#50#200# (1 mark , working should be shown) Minimum value for the variable Taker = 2 (½ Mark) Maximum value for the variable Taker = 3 (½ Mark)	2
(e)	75#15 50#5 750#75 (½ Mark for writing each correct value)	3
(f)	Output 12*63*73*15*93*10* (3 Marks for correct Output) (½ Mark for writing each correct value)	3
2(a)	Data Encapsulation: Wrapping up of data and functions together in a single unit is known as Data Encapsulation. In a class, we wrap up the data and functions together in a single unit. (½ Mark) Data Hiding: Keeping the data in private visibility mode of the class to prevent it from accidental change is known as Data Hiding. . (½ Mark) Any example. Explaining both (1 Mark)	2

(b)	An object is an identifiable entity with some characteristics and behavior. It represents an entity that can store data and its associated functions. (1 Mark) A class is a group of objects that share common properties and relationships. It represents a group of similar objects. (1 Mark)	2
3(a)	correct comparison between default arguments and function overloading (1 Mark each)	2

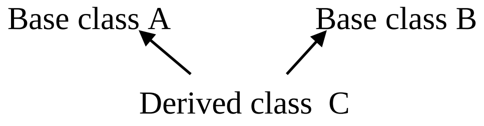
(b)	(i) callin(56); //function 4
	(ii) callin('p',77.2F); //function 2
	(iii) callin(83,77.33f); //function 3
	(iv)callin(4,66,’x’); //function 1
	(½ Mark each)
		2

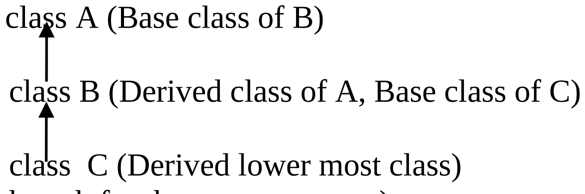
4(a)	Private Visibility Mode (1 Mark) The members in private visibility modes are not accessible to objects as well as derived classes Protected Visibility Mode (1 Mark) The members in protected visibility modes are not accessible to objects but are accessible in derived classes.	2

(b)	class ENVIRONMENT
	{ char City[20];
	int PMLevel;
	char Health[15];
	void AssignHealth();
	public :
	void In();
	void Out();
	};
	void ENVIRONMENT::AssignHealth()
	{ if (PMLevel<=50)
	strcpy(Health,"Healthy");
	else if (PMLevel<=100)
	strcpy (Health,"Moderate");
	else strcpy(Health,"Unhealthy");
	}
	void ENVIRONMENT::In() { gets(City); cin>>PMLevel;
	AssignHealth(); }
	void ENVIRONMENT::Out()
	{ cout<<City<<PMLevel<<Health<<endl; }
	
	(½ Mark for declaring class header correctly)
	(½ Mark for declaring data members correctly)
	(1 Mark for defining AssignHealth() correctly)
	(½ Mark for taking inputs of City and PMLevel in In())
	(½ Mark for invoking AssignHealth() inside In())
	(½ Mark for defining Out() correctly)
	(½ Mark for correctly closing class declaration with a semicolon ;)
		4

<table>
<tr><td>5(a)</td><td>(i) TV B("SONY", 20000.25) ; // any valid values

(ii) TV(TV &temp)
{ strcpy (company, temp.company) ;
price = temp.price ;
}
(1 Mark each for the correct answer)</td><td>2</td></tr>
<tr><td>(a)</td><td>OR
A copy constructor is an overloaded constructor in which an object of the same class is passed as reference parameter (1 Mark)

. class X { int a;
public: X()
{ a=0; }
X(X &ob) //copy constructor
{ a=ob.a; } };
Any example (1 Mark)</td><td></td></tr>
<tr><td>(b)</td><td>
<table>
<tr><td>Constructor</td><td>Destructor</td></tr>
<tr><td>Name of the constructor function is same as that of class</td><td>Name of the destructor function is same as that of class preceded by ~</td></tr>
<tr><td>Constructor functions are called automatically at the time of creation of the object</td><td>Destructor functions are called automatically when the scope of the object gets over</td></tr>
<tr><td>Constructor can be overloaded</td><td>Destructor cannot be overloaded</td></tr>
<tr><td>Constructor is used to initialize the data members of the class</td><td>Destructor is used to de- initialize the data members of the class</td></tr>
</table>
Correct Differences(Any 2)- (1 Mark each)</td><td>2</td></tr>
<tr><td>6(a)</td><td>In multiple inheritance a class is derived from two or more base classes.
Eg: 

In a multilevel inheritance a class is derived from an immediate base class.
Eg: 

class A (Base class of B)
class B (Derived class of A, Base class of C)
class C (Derived lower most class)

(1 Mark each for the correct answer)</td><td>2</td></tr>
</table>

<table>
<tr><td>(b)</td><td>(i)None of data members are accessible from objects belonging to class AUTHOR. (1 Mark)

(ii) Enter(), Show() (1 Mark)

(iii) Data members: Voucher_No, Sales_Date, Salary (½ Mark)

Member function:Sales_Entry(),Sales_Detail(),Enter(),Show(),Register(), Status() (½ Mark)

(iv) 66 bytes (1 Mark)</td><td>4</td></tr>
<tr><td>7(a)</td><td>void DispPorS ( )
{
ifstream File ("PLACES.TXT");
char STR[80];
while(File.getline(STR,80))
{
if(STR[0]=='P' || STR[0]=='S')
cout<<STR<<endl;
}
File.close();
}
(½ Mark for opening PLACES. TXT correctly)
(1 Mark for reading each Line (Whichever method adopted) from the file)
(1 Mark for checking lines starting with 'P' or 'S')
(½ Mark for displaying the lines)</td><td>3</td></tr>
<tr><td>(b)</td><td>void Economic()
{ GIFTS I;
  ifstream fin(“GIFTS.DAT”,ios::binary);
  while (fin.read((char *)&I,sizeof(I)))
 { if(I.GetCost()>2000)
      I.See();
}
 fin.close(); }
 (1 Mark for opening GIFTS.DAT correctly)
 (1 Mark for reading all records from the file)
(1 Mark for checking value of Cost > 2000)
 (1 Mark for displaying the desired items and closing the file)</td><td>4</td></tr>
<tr><td>(b)</td><td>OR
void Read_File( )
 { BUS B;
  ifstream Fin;</td><td></td></tr>
</table>

	Fin.open(“Bus.Dat”, ios::binary); while(Fin.read((char *) &B, sizeof(B))) { if(strcmp(B.EndTo(), “Mumbai”)==0) { B.show( ) ; } } Fin.close( ); } (1 Mark for opening Bus.dat correctly) (1 Mark for reading all records from the file) (1 Mark for checking destination is Mumbai) (1 Mark for displaying the desired items and closing the file)	
(c)	(i) File.seekg(-1 *sizeof(I) ,ios: :cur)); (1 Mark) (ii) File.write((char*)&I,sizeof(I)); (1 Mark)	2

(c)	OR
	seekp(): This function takes the file put pointer to the specified byte in a file.
	Eg: f.seekp(30); // It takes a pointer to 30th byte. (1 Mark)
	seekg(): This function takes the file get pointer to the specified byte in a file.
	Eg: f.seekg(30); // It takes a pointer to 30th byte. (1 Mark)	

8(a)	(½ Mark for function header) (1 Marks for the correct Logic to search an integer using binary search) (½ Mark for correct return statement)	2
(b)	(½ Mark for function header) (1 Mark for correct loop) (1½ Marks for the correct Logic for sorting)	3
(c)	(½ Mark for function header) (2½ Marks for the correct Logic)	3
(d)	void DISPMID(int A[][5],int R,int C) { for (int J=0;J<C;J++) cout<<A[R/2][J]<< “ “; cout<<endl; for (int I=0;I<R;I++) cout<<A[I][C/2]<< “ “; } (1 Mark for correct loop) (2 Marks for the correct Logic to display middle row and middle column)	3

(d)	OR
	void SWAPCOL(int A[][100], int M, int N)
	{int Temp, I;
	for(I=0; I<M; I++)
	{Temp = A [I][0] ;
	A[I][0] = A[I][N-l];
	A[I][N-l] = Temp;}}	

	(1 Mark for correct loop) (2 Mark for swapping the first column with last column correctly)	
(e)	Col-major Formula:- S[I][J]= B + W*[(I-Lr) + (J-Lc) *M] [1 Mark] W = size of each location in bytes = 4 Lr = Lower Bound of rows = 0 Lc = Lower Bound of columns = 0 M = Number of rows per column = 40 Address of S[I][J]=BaseAddress + W*[(I-Lr) + (J-Lc) *M] Address of S[15] [10] = BaseAddress+ 4[(15–0)+(10 - 0)*40] 7200= Base Address + 4 [415] Base Address = 7200 - 4 * 415 = 7200 - 1660 **= 5540** Address of S[20][15]= 5540 + 4 [(20 - 0) + (15 - 0) x 40] = 5540 + 4 x 620 = 5540 + 2480 **= 8020** 1 Mark for writing correct formula OR substituting formula with correct values) (1 Mark for correct step calculations) (1 Mark for final correct address)	3
(e)	OR Loc (ARR[I] [J]) along the row =BaseAddress + W [(I - LBR)*C + (J - LBC)] (where C is the number of columns, LBR = LBC = 0 LOC (ARR[10] [5]) = BaseAddress + W [I*C + J] 15000 = BaseAddress + 4[10*20 + 5] = BaseAddress + 4[200 + 5] = BaseAddress + 4 x 205 = BaseAddress + 820 BaseAddress = 15000-820 **= 14180** LOC (ARR[30] [10]) = 14180 + 4 [30 * 20 + 10] = 14180 + 4 * 610 = 14180 + 2440 **= 16620** 1 Mark for writing correct formula OR substituting formula with correct values) (1 Mark for correct step calculations) (1 Mark for final correct address)	
9(a)	(½ Mark for checking function header) (2½ Mark for logic to add a book information in stack)	3
(b)	(½ Mark for checking function header) (2½ Mark for logic deleting the product from the Queue)	3

(b)

OR

(½ Mark for checking function header)

(2½ Mark for logic to add a member in the Queue)

(c)

Steps	Element Scanned	Action	Stack Status	Result
1	(		(	
2	6	Push 6	(6	
3	10	Push 10	(6 10	
4	5	Push 5	(6 10 5	
5	+	Pop 5 & 10 Push 15	(6 15	10 + 5 = 15
6	*	Pop 15 & 6 Push 90	(90	6 * 15
7	18	Push 18	(90 18	
8	3	Push 3	(90 18 3	
9	/	Pop 18 & 3 Push 6	(90 6	18/3 = 6
10	-	Pop 90 & 6 Push 84	(84	90 - 6
11	)	Stack empty		84

Result = 84

(1 Marks for correct steps showing stack status)

(1 Mark for correct output)

2

(d)

A/(B+C)*D-E = (A / (B+C) * D - E)

Element	Stack of Operators	Postfix Expression
(	(	
A	(	A
/	(/	A
(	(/(	A
B	(/(	AB
+	(/(+	AB
C	(/(+	ABC
)	(/	ABC+
*	(*	ABC+/
D	(*	ABC+/D
-	(-	ABC+/D*
E	(-	ABC+/D*E
)		ABC+/D*E-

= **ABC+/D*E-**

(1 Mark for correctly converting till each operator)

(1 Mark to be given for writing correct answer)

2