|  |  |
| --- | --- |
| **Id** | 1 |
| Question | What does the following statement mean?   int (\*fp)(char\*) |
| A | pointer to a pointer |
| B | pointer to an array of chars |
| C | pointer to function taking a char\* argument and returns an int |
| D | function taking a char\* argument and returning a pointer to int |
| Answer | C |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 2 |
| Question | The operator used for dereferencing or indirection is \_\_\_\_ |
| A | \* |
| B | & |
| C | -> |
| D | –>> |
| Answer | A |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 3 |
| Question | Choose the right option     string\* x, y; |
| A | x is a pointer to a string, y is a string |
| B | y is a pointer to a string, x is a string |
| C | both x and y are pointer to string types |
| D | none of the mentioned |
| Answer | A |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 4 |
| Question | Which one of the following is not a possible state for a pointer? |
| A | hold the address of the specific object |
| B | point one past the end of an object |
| C | Zero |
| D | point to a byte |
| Answer | D |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 5 |
| Question | Which of the following is illegal? |
| A | int \*ip; |
| B | string s, \*sp = 0; |
| C | int i; double\* dp = &i; |
| D | int \*pi = 0; |
| Answer | D |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 6 |
| Question | #include <iostream>  using namespace std;  int main()  {  int a = 5, b = 10, c = 15;  int \*arr[ ] = {&a, &b, &c};  cout <<arr[1];  return 0;  } |
| A | 10 |
| B | 15 |
| C | 20 |
| D | Random number |
| Answer | D |
| Marks | 2 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 7 |
| Question | The correct statement for a function that takes pointer to a float, a pointer to a pointer to a char and returns a pointer to a pointer to a integer is |
| A | int \*\*fun(float\*\*, char\*\*) |
| B | int \*fun(float\*, char\*) |
| C | int \*\*\*fun(float\*, char\*\*) |
| D | int \*\*\*fun(\*float, \*\*char) |
| Answer | C |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 8 |
| Question | #include <iostream>  using namespace std;  int main()  {  char arr[20];  int i;  for(i = 0; i < 10; i++)  \*(arr + i) = 65 + i;  \*(arr + i) = '\0';  cout << arr;  return(0);  } |
| A | ABCDEFGHIJ |
| B | AAAAAAAAA |
| C | JJJJJJJJJJJJ |
| D | None |
| Answer | A |
| Marks | 2 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 9 |
| Question | #include <iostream>  using namespace std;  int main()  {  char \*ptr;  char Str[] = "abcdefg";  ptr = Str;  ptr += 5;  cout << ptr;  return 0;  } |
| A | fg |
| B | cdef |
| C | defg |
| D | abcd |
| Answer | A |
| Marks | 2 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 10 |
| Question | Which rule will not affect the friend function? |
| A | private and protected members of a class cannot be accessed from outside |
| B | private and protected member can be accessed anywhere |
| C | both a &b |
| D | None |
| Answer | A |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 11 |
| Question | Which keyword is used to declare the friend function? |
| A | Firend |
| B | friend |
| C | Classfriend |
| D | myfriend |
| Answer | B |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
|  | 12 |
| Question | #include <iostream>  using namespace std;  class Box  {  double width;  public:  friend void printWidth( Box box );  void setWidth( double wid );  };  void Box::setWidth( double wid )  {  width = wid;  }  void printWidth( Box box )  {  box.width = box.width \* 2;  cout << "Width of box : " << box.width << endl;  }  int main( )  {  Box box;  box.setWidth(10.0);  printWidth( box );  return 0;  } |
| A | 40 |
| B | 5 |
| C | 10 |
| D | 20 |
| Answer | D |
| Marks | 2 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 13 |
| Question | Pick out the correct statement. |
| A | A friend function may be a member of another class. |
| B | A friend function may not be a member of another class. |
| C | A friend function may or may not be a member of another class. |
| D | None of the mentioned |
| Answer | C |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 14 |
| Question | Where does keyword ‘friend’ should be placed? |
| A | function declaration |
| B | function definition |
| C | main function |
| D | None |
| Answer | A |
| Marks | 1 |
| Unit |  |

|  |  |
| --- | --- |
| **Id** | 15 |
| Question | #include <iostream>  using namespace std;  class sample  {  private:  int a, b;  public:  void test()  {  a = 100;  b = 200;  }  friend int compute(sample e1);  };  int compute(sample e1)  {  return int(e1.a + e1.b) - 5;  }  int main()  {  sample e;  e.test();  cout << compute(e);  return 0;  } |
| A | 100 |
| B | 200 |
| C | 300 |
| D | 295 |
| Answer | D |
| Marks | 2 |
| Unit |  |
| **Id** | 16 |
| Question | #include <iostream>  using namespace std;  class base  {  int val1, val2;  public:  int get()  {  val1 = 100;  val2 = 300;  }  friend float mean(base ob);  };  float mean(base ob)  {  return float(ob.val1 + ob.val2) / 2;  }  int main()  {  base obj;  obj.get();  cout << mean(obj);  return 0;  } |
| A | 200 |
| B | 150 |
| C | 100 |
| D | 300 |
| Answer |  |
| Marks | 2 |
| Unit |  |

|  |  |
| --- | --- |
| **Id** | 17 |
| Question | To which does the function pointer point to? |
| A | Variable |
| B | Constants |
| C | Function |
| D | absolute variables |
| Answer | C |
| Marks | 1 |
| Unit |  |

|  |  |
| --- | --- |
| **Id** | 18 |
| Question | What we will not do with function pointers? |
| A | allocation of memory |
| B | de-allocation of memory |
| C | both a &b |
| D | None |
| Answer | C |
| Marks | 1 |
| Unit |  |

|  |  |
| --- | --- |
| **Id** | 19 |
| Question | #include <iostream>  using namespace std;  int add(int first, int second)  {  return first + second + 15;  }  int operation(int first, int second, int (\*functocall)(int, int))  {  return (\*functocall)(first, second);  }  int main()  {  int a;  int (\*plus)(int, int) = add;  a = operation(15, 10, plus);  cout << a;  return 0;  } |
| A | 25 |
| B | 36 |
| C | 40 |
| D | 45 |
| Answer | C |
| Marks | 2 |
| Unit |  |

|  |  |
| --- | --- |
| **Id** | 20 |
| Question | #include <iostream>  using namespace std;  void func(int x)  {  cout << x ;  }  int main()  {  void (\*n)(int);  n = &func;  (\*n)( 2 );  n( 2 );  return 0;  } |
| A | 2 |
| B | 21 |
| C | 22 |
| D | 20 |
| Answer | C |
| Marks | 2 |
| Unit |  |

|  |  |
| --- | --- |
|  | 21 |
| Question | #include <iostream>  using namespace std;  int n(char, int);  int (\*p) (char, int) = n;  int main()  {  (\*p)('d', 9);  p(10, 9);  return 0;  }  int n(char c, int i)  {  cout << c << i;  return 0;  } |
| A | d9      9 |
| B | d9d9 |
| C | d9 |
| D | Compile time error |
| Answer | A |
| Marks | 2 |
| Unit |  |

|  |  |
| --- | --- |
| **Id** | 22 |
| Question | #include <iostream>  using namespace std;  int func (int a, int b)  {  cout << a;  cout << b;  return 0;  }  int main(void)  {  int(\*ptr)(char, int);  ptr = func;  func(2, 3);  ptr(2, 3);  return 0;  } |
| A | 2323 |
| B | 232 |
| C | 23 |
| D | Compile time error |
| Answer | D |
| Marks | 2 |
| Unit |  |

|  |  |
| --- | --- |
| **Id** | 23 |
| Question | What are the mandatory part to present in function pointers? |
| A | & |
| B | return values |
| C | Data types |
| D | None |
| Answer | C |
| Marks | 1 |
| Unit |  |

|  |  |
| --- | --- |
| **Id** | 24 |
| Question | What is meaning of following declaration?  int(\*ptr[5])(); |
| A | ptr is pointer to function. |
| B | ptr is array of pointer to function. |
| C | ptr is pointer to such function which return type is array. |
| D | ptr is pointer to array of function. |
| Answer | B |
| Marks | 1 |
| Unit |  |

|  |  |
| --- | --- |
| **Id** | 25 |
| Question | What is size of generic pointer in c? |
| A | 0 |
| B | 1 |
| C | 2 |
| D | Null |
| Answer | C |
| Marks | 1 |
| Unit |  |

|  |  |
| --- | --- |
| **Id** | 26 |
| Question | Void pointer can point to which type of objects? |
| A | Int |
| B | Float |
| C | Double |
| D | All |
| Answer | D |
| Marks | 1 |
| Unit |  |

|  |  |
| --- | --- |
| **Id** | 27 |
| Question | What does the following statement mean?       int (\*fp)(char\*) |
| A | pointer to a pointer |
| B | pointer to an array of chars |
| C | pointer to function taking a char\* argument and returns an int |
| D | function taking a char\* argument and returning a pointer to int |
| Answer | C |
| Marks | 1 |
| Unit |  |

|  |  |
| --- | --- |
| **Id** | 28 |
| Question | What is size of generic pointer in C++ (in 32-bit platform) ? |
| A | 2 |
| B | 4 |
| C | 8 |
| D | 0 |
| Answer | B |
| Marks | 1 |
| Unit |  |

|  |  |
| --- | --- |
| **Id** | 29 |
| Question | #include <iostream>  using namespace std;  int main()  {  int a[2][4] = {3, 6, 9, 12, 15, 18, 21, 24};  cout << \*(a[1] + 2) << \*(\*(a + 1) + 2) << 2[1[a]];  return 0;  } |
| A | 15 18 21 |
| B | 21 21 21 |
| C | 24 24 24 |
| D | Compile time error |
| Answer | B |
| Marks | 2 |
| Unit |  |

|  |  |
| --- | --- |
| **Id** | 30 |
| Question | #include <iostream>  using namespace std;  int main()  {  int i;  char \*arr[] = {"C", "C++", "Java", "VBA"};  char \*(\*ptr)[4] = &arr;  cout << ++(\*ptr)[2];  return 0;  } |
| A | ava |
| B | java |
| C | c++ |
| D | Compile time error |
| Answer | A |
| Marks | 2 |
| Unit |  |

|  |  |
| --- | --- |
| **Id** | 31 |
| Question | #include <iostream>  using namespace std;  int main()  {  int arr[] = {4, 5, 6, 7};  int \*p = (arr + 1);  cout << \*p;  return 0;  } |
| A | 4 |
| B | 5 |
| C | 6 |
| D | 7 |
| Answer | B |
| Marks | 2 |
| Unit |  |

|  |  |
| --- | --- |
| **Id** | 32 |
| Question | #include <iostream>  using namespace std;  int main()  {  int arr[] = {4, 5, 6, 7};  int \*p = (arr + 1);  cout << arr;  return 0;  } |
| A | 4 |
| B | 5 |
| C | Address of arr |
| D | 7 |
| Answer | C |
| Marks | 2 |
| Unit |  |

|  |  |
| --- | --- |
| **Id** | 33 |
| Question | #include <iostream>  using namespace std;  int main ()  {  int numbers[5];  int \* p;  p = numbers; \*p = 10;  p++; \*p = 20;  p = &numbers[2]; \*p = 30;  p = numbers + 3; \*p = 40;  p = numbers; \*(p + 4) = 50;  for (int n = 0; n < 5; n++)  cout << numbers[n] << ",";  return 0;  } |
| A | 10,20,30,40,50, |
| B | 1020304050 |
| C | Compile time error |
| D | Runtime error |
| Answer | A |
| Marks | 2 |
| Unit |  |

|  |  |
| --- | --- |
| **Id** | 34 |
| Question | #include <iostream>  using namespace std;  int main()  {  int arr[] = {4, 5, 6, 7};  int \*p = (arr + 1);  cout << \*arr + 9;  return 0;  } |
| A | 12 |
| B | 5 |
| C | 13 |
| D | Error |
| Answer | C |
| Marks | 2 |
| Unit |  |

|  |  |
| --- | --- |
| **Id** | 35 |
| Question | A void pointer cannot point to which of these? |
| A | methods in c++ |
| B | class member in c++ |
| C | all of the mentioned |
| D | None |
| Answer | D |
| Marks | 1 |
| Unit |  |

|  |  |
| --- | --- |
| **Id** | 36 |
| Question | #include <iostream>  using namespace std;  int func(void \*Ptr);  int main()  {  char \*Str = "abcdefghij";  func(Str);  return 0;  }  int func(void \*Ptr)  {  cout << Ptr;  return 0;  } |
| A | abcdefghij |
| B | address of string “abcdefghij” |
| C | Compile time |
| D | Run time error |
| Answer | B |
| Marks | 2 |
| Unit |  |

|  |  |
| --- | --- |
| **Id** | 37 |
| Question | #include <iostream>  using namespace std;  int main()  {  int \*p;  void \*vp;  if (vp == p);  cout << "equal";  return 0;  } |
| A | Equal |
| B | No output |
| C | Compile time error |
| D | Run time error |
| Answer | A |
| Marks | 2 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 38 |
| Question | #include <iostream>  using namespace std;  int main()  {  int n = 5;  void \*p = &n;  int \*pi = static\_cast<int\*>(p);  cout << \*pi << endl;  return 0;  } |
| A | 5 |
| B | 6 |
| C | Compile time error |
| D | Run time error |
| Answer | A |
| Marks | 2 |
| Unit |  |

|  |  |
| --- | --- |
| **Id** | 39 |
| Question | #include <iostream>  using namespace std;  int main()  {  int a = 5, c;  void \*p = &a;  double b = 3.14;  p = &b;  c = a + b;  cout << c << '\n' << p;  return 0;  } |
| A | 8, memory address |
| B | 8.14 |
| C | memory address |
| D | None |
| Answer | A |
| Marks | 2 |
| Unit |  |

|  |  |
| --- | --- |
| **Id** | 40 |
| Question | What we can’t do on a void pointer? |
| A | pointer arithemetic |
| B | pointer functions |
| C | Both |
| D | None |
| Answer | A |
| Marks | 2 |
| Unit |  |

|  |  |
| --- | --- |
| **Id** | 41 |
| Question | Which value we cannot assign to reference? |
| A | Integer |
| B | Floating |
| C | Unsigned |
| D | Null |
| Answer | D |
| Marks | 1 |
| Unit |  |

|  |  |
| --- | --- |
| **Id** | 42 |
| Question | #include <iostream>  using namespace std;  int main()  {  int a = 9;  int & aref = a;  a++;  cout << "The value of a is " << aref;  return 0;  } |
| A | 9 |
| B | 10 |
| C | 11 |
| D | Error |
| Answer | B |
| Marks | 2 |
| Unit |  |

|  |  |
| --- | --- |
| **Id** | 43 |
| Question | #include <iostream>  using namespace std;  void print (char \* a)  {  cout << a << endl;  }  int main ()  {  const char \* a = "Hello world";  print(const\_cast<char \*> (a) );  return 0;  } |
| A | Hello world |
| B | Hello |
| C | World |
| D | Compile time error |
| Answer | A |
| Marks | 2 |

|  |  |
| --- | --- |
| **Id** | 44 |
| Question | Identify the correct sentence regarding inequality between reference and pointer. |
| A | we can not create the array of reference. |
| B | we can create the Array of reference. |
| C | we can use reference to reference. |
| D | None |
| Answer | A |
| Marks | 1 |
| Unit |  |

|  |  |
| --- | --- |
| **Id** | 45 |
| Question | Which is used to tell the computer that where a pointer is pointing to? |
| A | Dereference |
| B | Reference |
| C | heap operations |
| D | None |
| Answer | A |
| Marks | 1 |
| Unit |  |

|  |  |
| --- | --- |
| Question | #include <iostream>  using namespace std;  int main()  {  int x;  int \*p;  x = 5;  p = &x;  cout << \*p;  return 0;  } |
| A | 5 |
| B | 10 |
| C | Memory address |
| D | None |
| Answer | A |
| Marks | 2 |
| Unit |  |

|  |  |
| --- | --- |
| **Id** | 47 |
| Question | #include <iostream>  using namespace std;  int main()  {  int x = 9;  int\* p = &x;  cout << sizeof(p);  return 0;  } |
| A | 4 |
| B | 2 |
| C | Depends on compiler |
| D | None |
| Answer | C |
| Marks | 2 |
| Unit |  |

|  |  |
| --- | --- |
| **Id** | 48 |
| Question | #include <iostream>  using namespace std;  int main()  {  double arr[] = {5.0, 6.0, 7.0, 8.0};  double \*p = (arr+2);  cout << \*p << endl;  cout << arr << endl;  cout << \*(arr+3) << endl;  cout << \*(arr) << endl;  cout << \*arr+9 << endl;  return 0;  } |
| A | 7     0xbf99fc98     8     5     14 |
| B | 7     8     0xbf99fc98     5     14 |
| C | 0xbf99fc98 |
| D | None |
| Answer | A |
| Marks | 2 |
| Unit |  |

|  |  |
| --- | --- |
| **Id** | 49 |
| Question | What does the dereference operator will return? |
| A | rvalue equivalent to the value at the pointer address. |
| B | lvalue equivalent to the value at the pointer address. |
| C | it will return nothing |
| D | None |
| Answer | B |
| Marks | 2 |
| Unit |  |

|  |  |
| --- | --- |
| **Id** | 50 |
| Question | Which operator is used in pointer to member function? |
| A | .\* |
| B | ->\* |
| C | Both a &b |
| D | None |
| Answer | C |
| Marks | 2 |
| Unit |  |

|  |  |
| --- | --- |
| **Id** | 51 |
| Question | #include <iostream>  using namespace std;  class Foo  {  public:  Foo(int i = 0){ \_i = i;}  void f()  {  cout << "Executed"<<endl;  }  private:  int \_i;  };  int main()  {  Foo \*p = 0;  p -> f();  } |
| A | Executed |
| B | Error |
| C | Run time error |
| D | None |
| Answer | A |
| Marks | 2 |
| Unit |  |

|  |  |
| --- | --- |
| **Id** | 52 |
| Question | Which is the best design choice for using pointer to member function? |
| A | Interface |
| B | Class |
| C | Structure |
| D | None |
| Answer | A |
| Marks | 2 |
| Unit |  |

|  |  |
| --- | --- |
| **Id** | 53 |
| Question | Virtual functions allow you to |
| A | create an array of type pointer-to-base class that can hold pointers to derived classes. |
| B | create functions that can never be accessed. |
| C | group objects of different classes so they can all be accessed by the same function code. |
| D | use the same function call to execute member functions of objects from different classes. |
| Answer | D |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 54 |
| Question | A pointer to a base class can point to objects of a derived class. |
| A | TRUE |
| B | FALSE |
| C |  |
| D |  |
| Answer | A |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 55 |
| Question | A pure virtual function is a virtual function that |
| A | causes its class to be abstract. |
| B | returns nothing. |
| C | is used in a base class. |
| D | A and C |
| Answer | D |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 56 |
| Question | An abstract class is useful when |
| A | no classes should be derived from it. |
| B | there are multiple paths from one derived class to another. |
| C | no objects should be instantiated from it. |
| D | you want to defer the declaration of the class. |
| Answer | C |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 57 |
| Question | A **friend**function can access a class’s private data without being a member of the class. |
| A | TRUE |
| B | FALSE |
| C |  |
| D |  |
| Answer | A |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 58 |
| Question | A **friend** function can be used to |
| A | mediate arguments between classes. |
| B | increase the versatility of an overloaded operator. |
| C | allow access to an unrelated class. |
| D | B and C |
| Answer | D |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 59 |
| Question | The keyword **friend** appears in |
| A | the class allowing access to another class. |
| B | the private section of a class. |
| C | the public section of a class. |
| D | All of the above |
| Answer | D |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 60 |
| Question | A static function |
| A | should be called when an object is destroyed. |
| B | is closely connected to an individual object of a class. |
| C | can be called using the class name and function name. |
| D | is used when a dummy object must be created. |
| Answer | C |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 61 |
| Question | An assignment operator might be overloaded to |
| A | help keep track of the number of identical objects. |
| B | assign a separate ID number to each object. |
| C | signal when assignment takes place. |
| D | All of the above |
| Answer | D |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 62 |
| Question | The user must always define the operation of the copy constructor. |
| A | TRUE |
| B | FALSE |
| C |  |
| D |  |
| Answer | B |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 63 |
| Question | The operation of the assignment operator and that of the copy constructor are |
| A | similar, except that the copy constructor creates a new object. |
| B | different, except that they both copy member data. |
| C | different, except that they both create a new object. |
| D | A and B |
| Answer | D |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 64 |
| Question | A copy constructor could be defined to copy only part of an object’s data. |
| A | TRUE |
| B | FALSE |
| C |  |
| D |  |
| Answer | A |
| Marks | 1 |
| Unit | III |
|  |  |

|  |  |
| --- | --- |
| Id | 65 |
| Question | The lifetime of a variable that is |
| A | local to a member function coincides with the lifetime of the function. |
| B | global coincides with the lifetime of a class. |
| C | nonstatic member data of an object coincides with the lifetime of the object. |
| D | A and C |
| Answer | D |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 66 |
| Question | There is no problem with returning the value of a variable defined as local within a member function so long as it is returned by value. |
| A | TRUE |
| B | FALSE |
| C |  |
| D |  |
| Answer | A |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 67 |
| Question | A copy constructor is invoked when |
| A | a function returns by value. |
| B | an argument is passed by value. |
| C | A and B |
| D | an argument is passed by reference. |
| Answer | C |
| Marks | 2 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 68 |
| Question | What does the **this**pointer point to? |
| A | Data member of the class |
| B | the object of which the function using it is a member |
| C | Member function |
| D | Base class |
| Answer | B |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 69 |
| Question | A pointer is |
| A | the address of a variable. |
| B | an indication of the variable to be accessed next. |
| C | a variable for storing addresses. |
| D | the data type of an address variable. |
| Answer | C |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 70 |
| Question | The expression **\*test**can be said to |
| A | refer to the contents of test. |
| B | dereference test. |
| C | refer to the value of the variable pointed to by test. |
| D | All of the above |
| Answer | D |
| Marks | 2 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 71 |
| Question | A pointer to void can hold pointers to \_\_\_\_\_\_\_\_\_\_ |
| A | int |
| B | float |
| C | char |
| D | Any data type |
| Answer | D |
| Marks | 1 |

|  |  |
| --- | --- |
| **Id** | 72 |
| Question | The type of variable a pointer points to must be part of the pointer’s definition so that |
| A | data types don’t get mixed up when arithmetic is performed on them. |
| B | pointers can be added to one another to access structure members. |
| C | the compiler can perform arithmetic correctly to access array elements. |
| D | A and C |
| Answer | D |
| Marks | 2 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 73 |
| Question | The first element in a string is |
| A | the name of the string. |
| B | the first character in the string. |
| C | the length of the string. |
| D | the name of the array holding the string. |
| Answer | b |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 74 |
| Question | The **new**operator |
| A | returns a pointer to a variable. |
| B | creates a variable called new. |
| C | obtains memory for a new variable. |
| D | A and C |
| Answer | D |
| Marks | 2 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 75 |
| Question | Definition for an array **arr**of 8 pointers that point to variables of type **float**is |
| A | \*float arr[8] |
| B | float\* arr[8]; |
| C | float pointer[8] |
| D | int \*ary[8] |
| Answer | B |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 76 |
| Question | The **delete** operator returns \_\_\_\_\_\_\_\_\_\_\_\_ to the operating system. |
| A | Memory that is no longer needed |
| B | Pointer |
| C | Object |
| D | Class |
| Answer | A |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 77 |
| Question | In a linked list |
| A | each link contains a pointer to the next link. |
| B | each link contains data or a pointer to data. |
| C | the links are stored in an array. |
| D | A and B |
| Answer | D |
| Marks | 2 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 78 |
| Question | If you wanted to sort many large objects or structures, it would be most efficient to |
| A | place them in an array and sort the array. |
| B | place pointers to them in an array and sort the array. |
| C | place them in a linked list and sort the linked list. |
| D | place references to them in an array and sort the array. |
| Answer | B |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 79 |
| Question | The contents of two pointers that point to adjacent variables of type **float**differ by \_\_\_\_\_ |
| A | 1 byte |
| B | 2 bytes |
| C | 3 bytes |
| D | 4 bytes |
| Answer | D |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 80 |
| Question | Which of the following is true about virtual functions in C++. |
| A | Virtual functions are functions that can be overridden in derived class with the same signature. |
| B | Virtual functions enable run-time polymorphism in a inheritance hierarchy. |
| C | If a function is 'virtual'in the base class, the most-derived class's implementation of the function is called according to the actual type of the object referred to, regardless of the declared type of the pointer or reference. In non-virtual functions, the functions are called according to the type of reference or pointer |
| D | All of the above |
| Answer | D |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 81 |
| Question | Predict the output of following C++ program.  #include<iostream>  using namespace std;  class Base {  public:  Base() { cout<<"Constructor: Base"<<endl; }  virtual ~Base() { cout<<"Destructor : Base"<<endl; }  };  class Derived: public Base {  public:  Derived() { cout<<"Constructor: Derived"<<endl; }  ~Derived() { cout<<"Destructor : Derived"<<endl; }  };  int main() {  Base \*Var = new Derived();  delete Var;  return 0;  } |
| A | Constructor: Base  Constructor: Derived  Destructor : Derived  Destructor : Base |
| B | Constructor: Base  Constructor: Derived  Destructor : Base |
| C | Constructor: Base  Constructor: Derived  Destructor : Derived |
| D | Constructor: Derived  Destructor : Derived |
| Answer | A |
| Marks | 2 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 82 |
| Question | Predict the output of following C++ program. Assume that there is no alignment and a typical implementation of virtual functions is done by the compiler.  #include <iostream>  using namespace std;    class A  {  public:  virtual void fun();  };    class B  {  public:  void fun();  };    int main()  {  int a = sizeof(A), b = sizeof(B);  if (a == b) cout <<"a == b";  else if (a >b) cout <<"a >b";  else cout <<"a <b";  return 0;  } |
| A | a>b |
| B | a==b |
| C | a<b |
| D | Compiler error |
| Answer | A |
| Marks | 2 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 83 |
| Question | Which of the following is FALSE about references in C++ |
| A | A reference must be initialized when declared |
| B | Once a reference is created, it cannot be later made to reference another object; it cannot be reset |
| C | References cannot be NULL |
| D | References cannot refer to constant value |
| Answer | D |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 84 |
| Question | #include <iostream>  using namespace std;    class A  {  public:  virtual void fun() { cout <<"A::fun() "; }  };    class B: public A  {  public:  void fun() { cout <<"B::fun() "; }  };    class C: public B  {  public:  void fun() { cout <<"C::fun() "; }  };    int main()  {  B \*bp = new C;  bp->fun();  return 0;  }  Which function will be called by statements bp->fun();? |
| A | A::fun() |
| B | B::fun() |
| C | C::fun() |
| D | Compiler error |
| Answer | C |
| Marks | 2 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 85 |
| Question | Which of the followings is/are automatically added to every class, if we do not write our own. |
| A | Copy Constructor |
| B | Assignment Operator |
| C | A constructor without any parameter |
| D | All of the above |
| Answer | D |
| Marks | 2 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 85 |
| Question | What is the output of following program?  #include<iostream>  using namespace std;  class Point {  Point() { cout <<"Constructor called"; }  };    int main()  {  Point t1;  return 0;  } |
| A | Compiler Error |
| B | Runtime Error |
| C | Constructor called |
| D | Segmentation Fault |
| Answer | A |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 86 |
| Question | What will be the output of following program?  #include <iostream>  using namespace std;    class Test  {  public:  Test() { cout <<"Hello from Test() "; }  } a;    int main()  {  cout <<"Main Started ";  return 0;  } |
| A | Main Started |
| B | Main Started Hello from Test() |
| C | Hello from Test() Main Started |
| D | Compiler Error: Global objects are not allowed |
| Answer | C |
| Marks | 2 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 87 |
| Question | Which rule will not affect the friend function |
| A | private &protected members of a class cannot be accessed from outside |
| B | private &protected member can be accessed anywhere |
| C | both a &b |
| D | none of these |
| Answer | A |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 88 |
| Question | which keyword is used to declare the friend function |
| A | Friend |
| B | Class Friend |
| C | My friend |
| D | all above |
| Answer | A |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 89 |
| Question | what is syntax of friend function? |
| A | Friend class1 Class2; |
| B | Friend class; |
| C | Friend class |
| D | none of these |
| Answer | D |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 90 |
| Question | what is output of the program?  #include<iostream>  using namespace std;  class Box  {  double width;  public:  friend void printWidth(Box box);  void setWidth(double wid);  };  void Box::setWidth(double wid)  {  width-=wid;  }  void printWidth(Box box)  {  box.width=box.width\*2;  cout<<"Width of box :"<<box.width<<endl;  }  int main()  {  Box box;  box.setWidth(10.0);  printWidth(box);  return 0;  } |
| A | 40 |
| B | 5 |
| C | 10 |
| D | 20 |
| Answer | D |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 91 |
| Question | pick out the correct statement. |
| A | A friend function may be member of another class |
| B | A friend function may not be member of another class |
| C | A friend function may or may not be member of another class |
| D | none of these |
| Answer | C |

|  |  |
| --- | --- |
| **Id** | 92 |
| Question | Where does keyword 'friend' should be placed? |
| A | Function declaration |
| B | Function definition |
| C | Main function |
| D | none of these |
| Answer | A |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 94 |
| Question | Which of the following type of class allows only one object of it to be created? |
| A | Virtual class |
| B | Abstract class |
| C | Singleton class |
| D | Friend class |
| Answer | C |
| Marks | 1 |
| Unit | 3 |

|  |  |
| --- | --- |
| **Id** | 95 |
| Question | Which of the following is not type of constructor? |
| A | Copy constructor |
| B | Friend constructor |
| C | Default constructor |
| D | Parameterized constructor |
| Answer | B |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 96 |
| Question | Which of the following statement is correct? |
| A | Base class pointer cannot point to derived class |
| B | Derived class pointer cannot point to base class |
| C | Pointer to derived class cannot be created |
| D | Pointer to base class cannot be created |
| Answer | B |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 97 |
| Question | Which of the following is not the member of class? |
| A | Static function |
| B | Friend function |
| C | Const function |
| D | Virtual function |
| Answer | B |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 98 |
| Question | Which of the following is not member of class? |
| A | Data hiding |
| B | Dynamic Typing |
| C | Dynamic binding |
| D | Dynamic loading |
| Answer | C |
| Marks | 1 |

|  |  |
| --- | --- |
| **Id** | 99 |
| Question | The operator used for dereferencing or indirection is\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| A | \* |
| B | & |
| C | -> |
| D | ->> |
| Answer | D |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 100 |
| Question | Choose the right option  string\* x, y |
| A | x is pointer to string, y is a string |
| B | y is pointer to string , x is a string |
| C | both x &y are pointer to string types |
| D | none of these |
| Answer | A |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 101 |
| Question | Which one of the following is not a possible state for a pointer? |
| A | hold the address of specific object |
| B | point one past the end of an object |
| C | Zero |
| D | point to tye |
| Answer | D |
| Marks | 1 |
| Unit | 3 |

|  |  |
| --- | --- |
| **Id** | 102 |
| Question | Which of the following is illegal? |
| A | int \*ip; |
| B | string s, \*sp=0; |
| C | int i;double \*dp=&i; |
| D | int \*pi=0; |
| Answer | C |
| Marks | 1 |
| Unit | 3 |

|  |  |
| --- | --- |
| **Id** | 103 |
| Question | what will happen in the code?  int a=100,b=200;  int \*p=&a, \*q=&b;  p=q; |
| A | b is assigned to a |
| B | p now points to b |
| C | a is assigned to b |
| D | q now points to a |
| Answer | B |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 104 |
| Question | what is output of this program?  #include<iostream>  using namespace std;  int main()  {  int a=5, b=10, c=15;  int \*arr[]= {&a, &b, &c};  cout<<arr[1];  return 0;  } |
| A | 5 |
| B | 10 |
| C | 15 |
| D | it will return some random number |
| Answer | D |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| Question | The correct statement for a function that takes pointer to a float , a pointer to a ponter to a char &return a pointer to a integer is |
| A | int\*\*fun(float\*\*, char\*\*) |
| B | int \*fun(float\*, char\*) |
| C | int \*\*\*fun(float\*, char\*\*) |
| D | int \*\*\*fun(\*float, \*\*char) |
| Answer | C |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 106 |
| Question | What is size of generic pointer in C++(in 32-bit platform)? |
| A | 2 |
| B | 4 |
| C | 8 |
| D | 0 |
| Answer | B |
| Marks | 1 |
| Unit | 3 |

|  |  |
| --- | --- |
| **Id** | 107 |
| Question | What is the output of this program?  #include<iostream>  using namespace std;  int main()  {  int a[2][4]={3,6,9,12,15,18,21,24};  cout<<\*(a[1] + 2)<<\*(\*(a+1)+2)<<2[1[a]];  return 0;  } |
| A | 15 18 21 |
| B | 21 21 21 |
| C | 24 24 24 |
| D | compile time error |
| Answer | B |
| Marks | 1 |
| Unit | 3 |

|  |  |
| --- | --- |
| **Id** | 108 |
| Question | Void pointer can point to which type of objects? |
| A | Int |
| B | Float |
| C | Double |
| D | all of above |
| Answer | D |
| Marks | 1 |
| Unit | 3 |

|  |  |
| --- | --- |
| **Id** | 109 |
| Question | When does the void pointer can be dereferenced? |
| A | when it doesn't point to any value |
| B | when it cast to another type of object |
| C | using delete keyword |
| D | none of above |
| Answer | B |
| Marks | 1 |
| Unit | 3 |

|  |  |
| --- | --- |
| **Id** | 110 |
| Question | The pointer can point to any variable that is not declared with which of these? |
| A | Const |
| B | Volatile |
| C | both a &b |
| D | Static |
| Answer | C |
| Marks | 1 |
| Unit | 3 |

|  |  |
| --- | --- |
| **Id** | 111 |
| Question | A void pointer can not point to which of these? |
| A | methods in C++ |
| B | class member in c++ |
| C | both a &b |
| D | none of these |
| Answer | B |
| Marks | 1 |
| Unit | 3 |

|  |  |
| --- | --- |
| **Id** | 112 |
| Question | what we can’t do on void pointer? |
| A | pointer arithmetic |
| B | pointer functions |
| C | both a &b |
| D | none of these |
| Answer | A |
| Marks | 1 |
| Unit | 3 |

|  |  |
| --- | --- |
| **Id** | 113 |
| Question | To which does the function pointer point to? |
| A | Variable |
| B | Constants |
| C | Function |
| D | absolute variables |
| Answer | C |
| Marks | 1 |
| Unit | 3 |

|  |  |
| --- | --- |
| **Id** | 114 |
| Question | What we will not do with function pointers? |
| A | Allocation of memory |
| B | De-allocation of memory |
| C | both a &b |
| D | none of these |
| Answer | C |
| Marks | 1 |
| Unit | 3 |

|  |  |
| --- | --- |
| **Id** | 115 |
| Question | Which of the following can be passed in function pointers? |
| A | Variables |
| B | data types |
| C | Functions |
| D | none of these |
| Answer | C |
| Marks | 1 |
| Unit | 3 |

|  |  |
| --- | --- |
| **Id** | 116 |
| Question | Which operators are used in free store? |
| A | New |
| B | Delete |
| C | both a &b |
| D | none of these |
| Answer | C |
| Marks | 1 |
| Unit | 3 |

|  |  |
| --- | --- |
| **Id** | 117 |
| Question | What type of class member is operator new? |
| A | Static |
| B | Dynamic |
| C | Const |
| D | Smart |
| Answer | A |
| Marks | 1 |
| Unit | 3 |

|  |  |
| --- | --- |
| **Id** | 118 |
| Question | linked lists are not suitable to for the implementation of\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| A | insertion sort |
| B | radix sort |
| C | polynomial manipulation |
| D | binary search |
| Answer | D |
| Marks | 1 |
| Unit | 3 |

|  |  |
| --- | --- |
| **Id** | 119 |
| Question | Run time polymorphism can be achieved with\_\_\_\_\_\_\_\_\_\_\_\_ |
| A | virtual base class |
| B | container class |
| C | virtual function |
| D | a &c |
| Answer | C |
| Marks | 1 |
| Unit | 3 |

|  |  |
| --- | --- |
| **Id** | 120 |
| Question | When a virtual function is redefine by the derived class, it is called\_\_\_\_\_\_ |
| A | Overloading |
| B | Overriding |
| C | Rewriting |
| D | all of the above |
| Answer | A |
| Marks | 1 |
| Unit | 3 |

|  |  |
| --- | --- |
| **Id** | 121 |
| Question | An abstract class is useful when |
| A | no classes should be derived from it. |
| B | there are multiple paths from one derived class to another. |
| C | no objects should be instantiated from it. |
| D | you want to defer the declaration of the class. |
| Answer | C |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| Id | 122 |
| Question | Use of virtual functions implies |
| A | Overloading |
| B | Overriding |
| C | Static binding |
| D | Dynamic binding |
| Answer | D |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| Id | 123 |
| Question | Which of the following type casts will convert an Integer variable named amount to a Double type? |
| A | (double) amount |
| B | |  |  | | --- | --- | |  | ( int to double) amount | |
| C | |  |  | | --- | --- | |  | int to double(amount) | |
| D | int (amount) to double |
| Answer | A |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| Id | 124 |
| Question | Pure virtual functions |
| A | Have to be redefined in the inherited class |
| B | Cannot have public access specification |
| C | Are mandatory for a virtual class |
| D | None of the above |
| Answer | A |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| Id | 125 |
| Question | A friend function to a class, C cannot access |
| A | Private data members and member functions |
| B | Public data members and member functions |
| C | Protected data members and member functions |
| D | The data members of the derived class of C |
| Answer | D |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 126 |
| Question | The function whose prototype is void getData(Item \*thing); receives |
| A | |  |  | | --- | --- | |  | a pointer to a structure | |
| B | a reference to a structure |
| C | a copy of a structure |
| D | None |
| Answer | A |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| Id | 127 |
| Question | The keyword friend does not appear in |
| A | The class allowing access to another class |
| B | The class desiring access to another class |
| C | The private section of a class |
| D | The public section of a class |
| Answer | C |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| Id | 128 |
| Question | What is the output of the following code char symbol[3]={‘a’,‘b’,‘c’}; for (int index=0; index<3; index++) cout <<symbol [index]; |
| A | a b c |
| B | “abc” |
| C | abc |
| D | ‘abc’ |
| Answer | C |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 129 |
| Question | Predict output of the following program  #include<iostream>  using namespace std;    class Base  {  public:      virtual void show() { cout<<" In Base \n"; }  };    class Derived: public Base  {  public:      void show() { cout<<"In Derived \n"; }  };    int main(void)  {      Base \*bp = new Derived;      bp->show();        Base &br = \*bp;      br.show();        return 0;  } |
| A | In Base  In Base |
| B | In Base  In Derived |
| C | In Derived  In Derived |
| D | In Derived  In Base |
| Answer | C |
| Marks | 2 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 130 |
| Question | Output of following program   |  | | --- | | #include<iostream>  using namespace std;    class Base  {  public:      virtual void show() { cout<<" In Base \n"; }  };    class Derived: public Base  {  public:      void show() { cout<<"In Derived \n"; }  };    int main(void)  {      Base \*bp, b;      Derived d;      bp = &d;      bp->show();      bp = &b;      bp->show();      return 0;  } | |
| A | In Base  In Base |
| B | In Base  In Derived |
| C | In Derived  In Derived |
| D | In Derived  In Base |
| Answer | D |
| Marks | 2 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 131 |
| Question | Which of the following is true about pure virtual functions?  1) Their implementation is not known in a class where they are declared.  2) If a class has a pure virtual function, then the class becomes abstract class and an instance of this class cannot be created. |
| A | Only 1 |
| B | Only 2 |
| C | Both |
| D | None |
| Answer | C |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 132 |
| Question | #include<iostream>  using namespace std;    class Base  {  public:      virtual void show() = 0;  };    int main(void)  {      Base b;      Base \*bp;      return 0;  } |
| A | There are compiler errors in lines "Base b;" and "Base bp;" |
| B | There is compiler error in line "Base b;" |
| C | There is compiler error in line "Base bp;" |
| D | No compilation error |
| Answer | B |
| Marks | 2 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 133 |
| Question | Predict the output of following program.   |  | | --- | | #include<iostream>  using namespace std;  class Base  {  public:      virtual void show() = 0;  };    class Derived : public Base { };    int main(void)  {      Derived q;      return 0;  } | |
| A | Compiler Error: there cannot be an empty derived class |
| B | Compiler Error: Derived is abstract |
| C | No compiler Error |
| D | None |
| Answer | B |
| Marks | 2 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 134 |
| Question | #include<iostream>  using namespace std;    class Base  {  public:      virtual void show() = 0;  };    class Derived: public Base  {  public:      void show() { cout<<"In Derived \n"; }  };    int main(void)  {      Derived d;      Base &br = d;      br.show();      return 0;  } |
| A | Compiler Error in line "Base &br = d;" |
| B | Empty output |
| C | In derived |
| D | None |
| Answer | C |
| Marks | 2 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 135 |
| Question | Can a constructor be virtual? Will the following program compile?   |  | | --- | | #include <iostream>  using namespace std;  class Base {  public:    virtual Base() {}  };  int main() {     return 0;  } | |
| A | Yes |
| B | No |
| C |  |
| D |  |
| Answer | B |
| Marks | 2 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 136 |
| Question | Can a destructor be virtual? Will the following program compile?   |  | | --- | | #include <iostream>  using namespace std;  class Base {  public:    virtual ~Base() {}  };  int main() {     return 0;  } | |
| A | Yes |
| B | No |
| C |  |
| D |  |
| Answer | A |
| Marks | 2 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 137 |
| Question | Predict the output  #include<iostream>  using namespace std;  class Base  {  public:      Base()    { cout<<"Constructor: Base"<<endl; }      virtual ~Base()   { cout<<"Destructor : Base"<<endl; }  };  class Derived: public Base {  public:      Derived()   { cout<<"Constructor: Derived"<<endl; }      ~Derived()  { cout<<"Destructor : Derived"<<endl; }  };  int main()  {      Base \*Var = new Derived();      delete Var;      return 0;  } |
| A | Constructor: Base  Constructor: Derived  Destructor : Derived  Destructor : Base |
| B | Constructor: Base  Constructor: Derived  Destructor : Base |
| C | Constructor: Base  Constructor: Derived  Destructor : Derived |
| D | Constructor: Derived  Destructor : Derived |
| Answer | A |
| Marks | 2 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 138 |
| Question | Can static functions be virtual? Will the following program compile?   |  | | --- | | #include<iostream>  using namespace std;    class Test  {     public:        virtual static void fun()  { }  }; | |
| A | Yes |
| B | No |
| C |  |
| D |  |
| Answer | B |
| Marks | 2 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 139 |
| Question | Predict the output of following C++ program. Assume that there is no alignment and a typical implementation of virtual functions is done by the compiler.   |  | | --- | | #include <iostream>  using namespace std;    class A  {  public:      virtual void fun();  };    class B  {  public:     void fun();  };    int main()  {      int a = sizeof(A), b = sizeof(B);      if (a == b) cout <<"a == b";      else if (a >b) cout <<"a >b";      else cout <<"a <b";      return 0;  } | |
| A | a>b |
| B | a==b |
| C | a<b |
| D | Compile time error |
| Answer | A |
| Marks | 2 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 140 |
| Question | #include <iostream>  using namespace std;    class A  {  public:      virtual void fun() { cout <<"A::fun() "; }  };    class B: public A  {  public:     void fun() { cout <<"B::fun() "; }  };    class C: public B  {  public:     void fun() { cout <<"C::fun() "; }  };    int main()  {      B \*bp = new C;      bp->fun();      return 0;  } |
| A | a::fun() |
| B | b::fun() |
| C | c::fun() |
| D | None |
| Answer | C |
| Marks | 2 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 141 |
| Question | Predict the output of following C++ program   |  | | --- | | #include<iostream>  using namespace std;    class Base  {  public:      virtual void show() { cout<<" In Base \n"; }  };    class Derived: public Base  {  public:      void show() { cout<<"In Derived \n"; }  };    int main(void)  {      Base \*bp = new Derived;      bp->Base::show();  // Note the use of scope resolution here      return 0;  } | |
| A | In Base |
| B | In derived |
| C | Compile time error |
| D | Runtime error |
| Answer | A |
| Marks | 2 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 142 |
| Question | Which of the following is true about this pointer? |
| A | It is passed as a hidden argument to all function calls |
| B | It is passed as a hidden argument to all non-static function calls |
| C | It is passed as a hidden argument to all static functions |
| D | None |
| Answer | B |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 143 |
| Question | What is the use of this pointer? |
| A | When local variable’s name is same as member’s name, we can access member using this pointer. |
| B | To return reference to the calling object |
| C | Can be used for chained function calls on an object |
| D | All |
| Answer | D |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 144 |
| Question | Predict the output of following C++ program.   |  | | --- | | #include<iostream>  using namespace std;    class Test  {  private:    int x;  public:    Test(int x = 0) { this->x = x; }    void change(Test \*t) { this = t; }    void print() { cout <<"x = " <<x <<endl; }  };    int main()  {    Test obj(5);    Test \*ptr = new Test (10);    obj.change(ptr);    obj.print();    return 0;  } | |
| A | X=5 |
| B | X=10 |
| C | Compile time error |
| D | Run time error |
| Answer | C |
| Marks | 2 |
| Unit | III |

|  |  |
| --- | --- |
| Id | 145 |
| Question | A static data member is given a value |
| A | Within the class definition |
| B | |  |  | | --- | --- | |  | Outside the class definition | |
| C | When the program is exeuted |
| D | Never |
| Answer | D |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| Id | 146 |
| Question | A function call mechanism that passes arguments to a function by passing a copy of the values of the arguments is \_\_\_\_\_\_\_\_\_\_ |
| A | Call by name |
| B | Call by value |
| C | Call by reference |
| D | Call by value result |
| Answer | B |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 147 |
| Question | A ……………. takes a reference to an object of the same class as itself as an argument. |
| A | Reference constructor |
| B | Copy Constructor |
| C | Self Constructor |
| D | None of the above |
| Answer | B |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 148 |
| Question | Automatic initialization of object is carried out using a special member function called |
| A | Friend |
| B | Casting |
| C | Reference Parameter |
| D | Constructor |
| Answer | D |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 149 |
| Question | Which of the following condition is true for an object used as a function argument?  i) A copy of the entire objects is passed to the function.  ii) Only the address of the object is transferred to the function. |
| A | Only i |
| B | Only ii |
| C | Both I &ii |
| D | None |
| Answer | C |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 150 |
| Question | Which of the following parameter passing mechanism is/are supported by C++ not C |
| A | Pass by value |
| B | Pass by reference |
| C | Pass by value result |
| D | All of above |
| Answer | B |
| Marks | 1 |
| Unit | III |

|  |  |
| --- | --- |
| **Id** | 1 |
| Question | What is meaning of template parameter? |
| A | It is used to pass a type as argument |
| B | Used to evaluate a type |
| C | It can of no return type |
| D | None of the mentioned |
| Answer | A |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 2 |
| Question | \_\_\_\_\_\_Keyword is used in template. |
| A | Class |
| B | Typename |
| C | Both a and b |
| D | Using |
| Answer | C |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 3 |
| Question | What is scope of template parameter? |
| A | Inside a block only |
| B | Inside the class only |
| C | Throughout program |
| D | All of the above |
| Answer | A |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 4 |
| Question | Function overloading is also similar to which of the following |
| A | Operator overloading |
| B | Destructor overloading |
| C | Constructor overloading |
| D | Virtual function |
| Answer | B |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 5 |
| Question | Generic programming is approach of\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_which are applicable for all types |
| A | Generalised algorithm |
| B | Pseude algorithm |
| C | Both a and b |
| D | None of the above |
| Answer | A |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 6 |
| Question | Template are of types |
| A | Function template |
| B | Class template |
| C | Both a and b |
| D | None of the above |
| Answer | C |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 7 |
| Question | Class template can be created using\_\_\_\_\_\_\_\_syntax. |
| A | Template<class T>class classname |
| B | Template<class T1,class T2> class classname |
| C | Both a and b |
| D | None of the above mentioned |
| Answer | C |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 8 |
| Question | Syntax for creating a function template is |
| A | Template<typename t>returntype function name |
| B | Template<class T> returntype function name |
| C | Both a and b |
| D | None of the above mentioned |
| Answer | C |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 9 |
| Question | Pick up the correct statement  i)template allow us to define generic classes and functions  ii)template support generic programming  iii)function template overloading is possible |
| A | i only |
| B | i and ii only |
| C | ii and iii only |
| D | i, ii and iii |
| Answer | D |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 10 |
| Question | Template function can be overloaded |
| A | True |
| B | False |
| C |  |
| D |  |
| Answer | A |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 11 |
| Question | Why we use :: template-template parameter? |
| A | binding |
| B | rebinding |
| C | both a &b |
| D | none of these |
| Answer | C |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 12 |
| Question | Which of the things does not require instantiation? |
| A | functions |
| B | non virtual member function |
| C | member class |
| D | all of the mentioned |
| Answer | D |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 13 |
| Question | A template provides a convenient way to make a family of |
| A | variables. |
| B | functions |
| C | classes |
| D | B and C |
| Answer | D |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 14 |
| Question | Templates automatically create different versions of a function, depending on user input. |
| A | TRUE |
| B | FALSE |
| C |  |
| D |  |
| Answer | B |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 15 |
| Question | A template class |
| A | is designed to be stored in different containers. |
| B | works with different data types. |
| C | generates objects which must all be identical. |
| D | generates classes with different numbers of member functions. |
| Answer | B |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 16 |
| Question | There can be more than one template argument. |
| A | TRUE |
| B | FALSE |
| C |  |
| D |  |
| Answer | A |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 17 |
| Question | Actual code for a template function is generated when |
| A | the function declaration appears in the source code. |
| B | the function definition appears in the source code. |
| C | a call to the function appears in the source code. |
| D | the function is executed at runtime. |
| Answer | C |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 18 |
| Question | An exception is typically caused by |
| A | the programmer who writes an application’s code. |
| B | the creator of a class who writes the class member functions. |
| C | a runtime error. |
| D | an operating system malfunction that terminates the program. |
| Answer | C |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 19 |
| Question | Statements that might cause an exception must be part of a **catch** block. |
| A | TRUE |
| B | FALSE |
| C |  |
| D |  |
| Answer | B |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 20 |
| Question | Exceptions are thrown |
| A | from the catch block to the try block. |
| B | from a throw statement to the try block. |
| C | from the point of the error to a catch block. |
| D | from a throw statement to a catch block. |
| Answer | D |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 21 |
| Question | A statement that throws an exception does not need to be located in a try block. |
| A | TRUE |
| B | FALSE |
| C |  |
| D |  |
| Answer | B |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 22 |
| Question | The following is/are errors for which an exception would typically be thrown: |
| A | An excessive amount of data threatens to overflow an array. |
| B | new cannot obtain the requested memory. |
| C | A power failure shuts down the system. |
| D | A and B |
| Answer | D |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 23 |
| Question | Additional information sent when an exception is thrown may be placed in |
| A | the throw keyword. |
| B | the function that caused the error. |
| C | the catch block. |
| D | an object of the exception class. |
| Answer | D |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 24 |
| Question | A program can continue to operate after an exception has occurred. |
| A | TRUE |
| B | FALSE |
| C |  |
| D |  |
| Answer | A |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 25 |
| Question | What is the output of following program?  #include <iostream> using namespace std; int main() { int x = -1; try {  cout <<"Inside try \n";  if (x <0)  {  throw x;  cout <<"After throw \n";  } } catch (int x ) {  cout <<"Exception Caught \n"; }  cout <<"After catch \n"; return 0; } |
| A | Inside try  Exception Caught  After throw  After catch |
| B | Inside try  Exception Caught  After catch |
| C | Inside try  Exception Caught |
| D | Inside try  After throw  After catch |
| Answer | B |
| Marks | 2 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 26 |
| Question | What is the advantage of exception handling?  1) Remove error-handling code from the software's main line of code.  2) A method writer can chose to handle certain exceptions and delegate others to the caller.  3) An exception that occurs in a function can be handled anywhere in the function call stack. |
| A | Only 1 |
| B | 1, 2 and 3 |
| C | 1 and 3 |
| D | 1 and 2 |
| Answer | B |
| Marks | 2 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 27 |
| Question | What should be put in a try block?  1. Statements that might cause exceptions  2. Statements that should be skipped in case of an exception |
| A | Only 1 |
| B | Only 2 |
| C | Both 1 and 2 |
| D | None of the above |
| Answer | C |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 28 |
| Question | What is the output of following program  #include<iostream>  using namespace std;    class Base {};  class Derived: public Base {};  int main()  {  Derived d;  try {  throw d;  }  catch(Base b) {  cout<<"Caught Base Exception";  }  catch(Derived d) {  cout<<"Caught Derived Exception";  }  return 0;  } |
| A | Caught Derived Exception |
| B | Caught Base Exception |
| C | Caught Derived Exception  Caught Base Exception |
| D | Compiler Error |
| Answer | B |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 29 |
| Question | What is the output of following program?  #include <iostream>  using namespace std;    int main()  {  try  {  throw 10;  }  catch (...)  {  cout <<"default exception\n";  }  catch (int param)  {  cout <<"int exception\n";  }    return 0;  } |
| A | default exception |
| B | int exception |
| C | default exception  int exception |
| D | Compiler Error |
| Answer | D |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 30 |
| Question | What is the output of following program?  #include <iostream>  using namespace std;    class Test {  public:  Test() { cout <<"Constructing an object of Test "<<endl; }  ~Test() { cout <<"Destructing an object of Test " <<endl; }  };    int main() {  try {  Test t1;  throw 10;  } catch(int i) {  cout <<"Caught "<<i <<endl;  }  } |
| A | Caught 10 |
| B | Constructing an object of Test  Caught 10 |
| C | Constructing an object of Test  Destructing an object of Test  Caught 10 |
| D | Compiler Errror |
| Answer | C |
| Marks | 2 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 31 |
| Question | What happens in C++ when an exception is thrown and not caught anywhere like following program?  #include <iostream>  using namespace std;    int fun() throw (int)  {  throw 10;  }    int main() {  fun();  return 0;  } |
| A | Compiler error |
| B | Abnormal program termination |
| C | Program doesn't print anything and terminates normally |
| D | None of the above |
| Answer | B |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 32 |
| Question | Which alternative can replace the throw statement ? |
| A | Exit |
| B | For |
| C | Break |
| D | Return |
| Answer | D |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 33 |
| Question | Which of the following keyword can not be appered inside the class? |
| A | Virtual |
| B | Static |
| C | Template |
| D | Friend |
| Answer | C |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 34 |
| Question | What is template? |
| A | Template is formula for creating a generic class |
| B | Template is used to manipulate class |
| C | Template is used for creating functions |
| D | None of these |
| Answer | A |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 35 |
| Question | Select the correct syntax of template: |
| A | Template |
| B | Template<> |
| C | Temp |
| D | None of these |
| Answer | B |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 36 |
| Question | A class is generated from template class is called \_\_\_\_\_\_\_. |
| A | inherited class |
| B | derived class |
| C | generated class |
| D | subclass |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 37 |
| Question | \_\_\_\_\_\_\_\_ is useful when template of template is used? |
| A | Friend function |
| B | Static function |
| C | Typedef |
| D | Inheritance |
| Answer | C |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 38 |
| Question | Which of the C++ feature allows you to create classes that are dynamic for using data types? |
| A | Templates |
| B | Inheritance |
| C | Polymorphism |
| D | Information hiding |
| Answer | A |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 39 |
| Question | A function template means \_\_\_\_\_\_\_. |
| A | creating a function having exact type |
| B | creating a function without having to specify exact type |
| C | both a and b |
| D | none of these |
| Answer | B |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 40 |
| Question | Which of the following is used to describe the function using placeholder type? |
| A | Template type parameter |
| B | Template parameter |
| C | Template type |
| D | None of these |
| Answer | A |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 41 |
| Question | String template is used \_\_\_\_\_. |
| A | to replace a string. |
| B | to replace a string with another string |
| C | to delete a string |
| D | none of these |
| Answer | B |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 42 |
| Question | Maximum number of template argument in function template is \_\_\_\_\_\_\_. |
| A | two |
| B | three |
| C | four |
| D | many |
| Answer | D |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 43 |
| Question | Template function must have |
| A | one or more than one argument |
| B | zero argument |
| C | only one argument |
| D | at least two arguments |
| Answer | A |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 44 |
| Question | Template function must have at least \_\_\_\_\_\_\_\_ generic data type. |
| A | zero |
| B | one |
| C | two |
| D | none of these |
| Answer | B |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 45 |
| Question | Templates provide way of abstracting \_\_\_\_\_\_ information. |
| A | type |
| B | data |
| C | method |
| D | access |
| Answer | A |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 46 |
| Question | If you create instantiation of a class template with an int and then create a second instantiation with a double then |
| A | once the function is used for one data type it becomes unavailable for other type |
| B | you can not perform this kind of operation in C++ |
| C | you must precede each function call with the word int or double |
| D | none of these |
| Answer | C |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 47 |
| Question | If templates were removed from C++,Which of the following will be true?  I. Some algorithms could no longer be implemented  II. Any particular algorithms could still be implemented but often less elegantly. |
| A | Only I is true |
| B | Only II is true |
| C | Both I and II is true |
| D | None of these |
| Answer | D |
| Marks | 1 |
| Unit | 4 |

|  |  |
| --- | --- |
| **Id** | 48 |
| Question | In the template <class T>declaration of T stands for \_\_\_\_\_\_\_\_. |
| A | integer data type |
| B | arbitary class |
| C | generic data types |
| D | none of these |
| Answer | C |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 49 |
| Question | What is the meaning of template parameter? |
| A | It is used to pass a type as argument |
| B | It is used to evalute a type |
| C | It has no return type |
| D | None of these |
| Answer | A |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 50 |
| Question | What can be passed by non-type template parameter during compile time? |
| A | Int |
| B | Double |
| C | Char |
| D | constant expression |
| Answer | D |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 51 |
| Question | Choose the correct statement from the following: |
| A | Template function will take long time to execute |
| B | Template functions are written when you want to have only one code for many different types |
| C | due to template function the duplicate code will get increased |
| D | None of these |
| Answer | B |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 52 |
| Question | How many types of templates are there in c++? |
| A | Two |
| B | Three |
| C | Four |
| D | None Of These |
| Answer | A |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 53 |
| Question | What is the task of compiler while handling template? |
| A | type association |
| B | Portability |
| C | code elimination |
| D | all of the above |
| Answer | C |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 54 |
| Question | What should be the name of the parameter that the template should take? |
| A | same as class |
| B | same as function |
| C | same as template |
| D | none of these |
| Answer | C |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 55 |
| Question | Which keyword can be used with template? |
| A | Typename |
| B | operator |
| C | both a and b |
| D | None of these |
| Answer | A |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 56 |
| Question | Which of the following describes a difference between template function and  template class in c++? |
| A | The compiler determines the type of a template function's arguments, but  the types of template classes must be stated explicitly when declaring objects |
| B | template functions cannot be defined for user-defined types, but template classes can |
| C | template classes cannot be defined for user-defined types,but  templatefunctions can. |
| D | None Of These |
| Answer | A |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 57 |
| Question | What is the validity of templet parameter? |
| A | Inside the class |
| B | Inside the block |
| C | whole program |
| D | None of these |
| Answer | B |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 58 |
| Question | Which of the following does not required installation ? |
| A | Non virtual member function |
| B | Member class |
| C | Function |
| D | All of above |
| Answer | D |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 59 |
| Question | Which keyword is used to handle the exception ? |
| A | Try |
| B | Catch |
| C | Throw |
| D | Exception |
| Answer | B |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 60 |
| Question | What is the use of the keyword finally ? |
| A | It is used at the start of the program for handling all the exceptions |
| B | It is used at the end of the program to handle all the exceptions |
| C | It can be used anywhere in the program to handle all the exceptions |
| D | None of these |
| Answer | B |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 61 |
| Question | Which of the following most preferred way of throwing and handling exception? |
| A | Throw by value and catch by reference |
| B | Throw by reference and catch by value |
| C | Throw by value and catch by value |
| D | None of these |
| Answer | A |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 62 |
| Question | Which of the following is the most general exception handler that catches exception of any type? |
| A | Catch(std::exception) |
| B | Catch(std:any\_exception) |
| C | Catch(...) |
| D | Catch() |
| Answer | C |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 63 |
| Question | Which of the following causes an exception |
| A | Missing parenthesis in main() |
| B | Calling a function which is not present |
| C | A syntax error |
| D | a run time error |
| Answer | D |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 64 |
| Question | Which block should be placed after try block ? |
| A | Throw |
| B | Catch |
| C | both a or b |
| D | none of these |
| Answer | C |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 65 |
| Question | Choose the correct statement |
| A | Exception are not suitable for critical points in the program |
| B | Exception are suitable for critical points in the program |
| C | Both a&b |
| D | None of these |
| Answer | A |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 66 |
| Question | In C++ program handling, a try block must be followed by \_\_\_\_\_catch blocks |
| A | exactly one |
| B | one or more |
| C | exactly two |
| D | none of these |
| Answer | B |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 67 |
| Question | The process of handling the actual exception occurs \_\_\_\_\_\_\_\_\_ |
| A | inside the program |
| B | outside the program |
| C | both a &b |
| D | none of these |
| Answer | B |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 68 |
| Question | Which of the following is used to check the error in the block? |
| A | Try |
| B | Throw |
| C | Catch |
| D | None of these |
| Answer | A |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 69 |
| Question | What should be present when throwing object ? |
| A | Constructor |
| B | Destructor |
| C | copy constructor |
| D | none of these |
| Answer | C |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 70 |
| Question | For handling the exception in C++ \_\_\_\_\_\_\_ are used |
| A | catch handlers |
| B | exception handlers |
| C | Pointers |
| D | none of these |
| Answer | B |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 71 |
| Question | For handling the exceptions in C++ \_\_\_\_\_\_\_\_\_ is used . |
| A | handler function |
| B | terminate function |
| C | both a &b |
| D | none of these |
| Answer | B |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 72 |
| Question | How many parameters does the throw expression can have ? |
| A | 0 |
| B | 1 |
| C | 2 |
| D | 3 |
| Answer | B |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 73 |
| Question | What kind of exceptions are used in C++ |
| A | Handled |
| B | Unhandled |
| C | Static |
| D | Dynamic |
| Answer | B |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 74 |
| Question | What will happen when exception is uncaught? |
| A | Arise an error |
| B | program will run |
| C | execute in a loop |
| D | none of these |
| Answer | A |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 75 |
| Question | Choose the correct statement |
| A | A function can throw any type of exception |
| B | a function can throw an exception of certain type only |
| C | A exception can't throw any type of exception |
| D | none of these |
| Answer | B |
| Marks | 1 |
| Unit | 4 |

|  |  |
| --- | --- |
| **Id** | 76 |
| Question | What fuunction will be called when we have uncaught exception? |
| A | Catch |
| B | Throw |
| C | Terminate |
| D | none of these |
| Answer | C |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 77 |
| Question | What will happen when a programs throws any other of exception other than specified ? |
| A | still execute |
| B | Terminate |
| C | raise an error |
| D | none of these. |
| Answer | C |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 78 |
| Question | Which statement is used to catch all types of exceptions? |
| A | catch() |
| B | catch(Test t) |
| C | catch |
| D | none of these |
| Answer | D |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 79 |
| Question | Which keyword can be used as a template |
| A | Exception |
| B | Typename |
| C | both a & b |
| D | Function |
| Answer | B |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 80 |
| Question | An Exception is thrown using \_\_\_\_\_\_\_\_\_\_\_\_\_keyword in cpp |
| A | Throws |
| B | Throw |
| C | Threw |
| D | Thrown |
| Answer | B |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 81 |
| Question | Which parameter is legal for non-type template? |
| A | pointer to member |
| B | object |
| C | class |
| D | none of these |
| Answer | A |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 82 |
| Question | Which of the things does not require instantiation? |
| A | functions |
| B | Non virtual member function |
| C | member class |
| D | all of these |
| Answer | D |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 83 |
| Question | Which of the following permits function overloading on c++? |
| A | Data Type |
| B | Number of arguments |
| C | A &B both |
| D | none of these |
| Answer | C |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 84 |
| Question | Function overloading is also similar to which of the following? |
| A | Operator Overloading |
| B | Constructer overloading |
| C | Destructor overloading |
| D | none of the above |
| Answer | B |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 85 |
| Question | Which is dependent on template parameter |
| A | base class |
| B | abstract class |
| C | method |
| D | none of the above |
| Answer | A |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 87 |
| Question | How to declare a template? |
| A | Tem |
| B | Temp |
| C | Template<> |
| D | none of these |
| Answer | C |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 88 |
| Question | What may be the name of parameter that the template should take? |
| A | same as template |
| B | same as class |
| C | same as function |
| D | none of these |
| Answer | A |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 89 |
| Question | Which is used to handle the exceptions in c++? |
| A | catch handler |
| B | handler |
| C | exception handler |
| D | all of these |
| Answer | B |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 90 |
| Question | Which is called on allocating the memory for array of objects? |
| A | Function |
| B | Method |
| C | Destructor |
| D | Constructor |
| Answer | D |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 91 |
| Question | Which value is placed in the base class? |
| A | Inherited value |
| B | Derived value |
| C | Default type values |
| D | Both a and b |
| Answer | C |
| Marks | IV |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 92 |
| Question | Which is used to get the input during runtime? |
| A | cout |
| B | cin |
| C | Template |
| D | All of the above |
| Answer | B |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 93 |
| Question | \_\_\_\_\_\_\_\_\_\_is used to perform the generic programming. |
| A | Class |
| B | Template |
| C | Function |
| D | Inheritance |
| Answer | All of the above |
| Marks | B |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 94 |
| Question | A template can be considered as a kind of macros |
| A | True |
| B | False |
| C |  |
| D |  |
| Answer | A |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 95 |
| Question | We can not define more than 2 placeholder in class/function template. |
| A | False |
| B | True |
| C |  |
| D |  |
| Answer | A |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 96 |
| Question | When template is defined with parameter that would be replaced by specified \_\_\_\_\_\_\_at the time of actual use of class or function. |
| A | Keyword |
| B | Operator |
| C | Datatype |
| D | None of the above mentioned |
| Answer | C |
| Marks | IV |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 97 |
| Question | Templates sometimes called as \_\_\_\_\_\_\_\_\_\_\_ |
| A | Parameterized classes |
| B | Parameterized function |
| C | Both a and b |
| D | None of the above mentioned |
| Answer | C |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 98 |
| Question | Exceptions are of type |
| A | Synchronous |
| B | Asynchronous |
| C | Both a and b |
| D | None of the above mentioned |
| Answer | C |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 99 |
| Question | “out-of-range”, “overflow” are the type of exceptions |
| A | Asynchronous |
| B | Synchronous |
| C | Default |
| D | None of the above |
| Answer | B |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 100 |
| Question | The most type of error--------. |
| A | Logical error |
| B | Syntactic error |
| C | Both a and b |
| D | Class |
| Answer | C |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 101 |
| Question | Run time error is known as \_\_\_\_\_\_ |
| A | Logical error |
| B | Syntactic error |
| C | Exception |
| D | All of the above mentioned |
| Answer | C |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 102 |
| Question | How the exception is throw |
| A | throw exception |
| B | throw(exception) |
| C | throw |
| D | All of the above |
| Answer | D |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 103 |
| Question | Can we throw exception more than one time |
| A | True |
| B | False |
| C |  |
| D |  |
| Answer | A |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 104 |
| Question | Which statement we have to use rethrowing exception |
| A | throw(exception) |
| B | Throw |
| C | Both a and b |
| D | None of the above mentioned |
| Answer | B |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 105 |
| Question | Exception can be handle if\_\_\_\_\_\_\_ |
| A | Throwing argument is match with catch block |
| B | Throwing argument is not match with catch block |
| C | Exception is not thrown |
| D | None of the abve |
| Answer | A |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 106 |
| Question | With this concept same algorithm can be used for different data types |
| A | Procedure oriented paradigm |
| B | Generic programming |
| C | Both a and b |
| D | None of the above |
| Answer | B |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 106 |
| Question | Template is a way creating generalize functions and classes which are applicable for all data types |
| A | False |
| B | True |
| C |  |
| D |  |
| Answer | B |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 107 |
| Question | Class template is applicable for \_\_\_. |
| A | For function only |
| B | For that class only |
| C | Both a and b |
| D | None of the above mentioned |
| Answer | B |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 108 |
| Question | Function template is applicable for\_\_\_\_\_\_\_\_\_. |
| A | For function only |
| B | For that class only |
| C | Both a and b |
| D | None of the above mentioned |
| Answer | A |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 109 |
| Question | How many kinds of parameters are there in c++ |
| A | 1 |
| B | 2 |
| C | 3 |
| D | 4 |
| Answer | C |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 110 |
| Question | Which type of program is recommended to include in try block |
| A | Static memory allocation |
| B | Dynamic memory allocation |
| C | Const reference |
| D | Pointer |
| Answer | B |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 111 |
| Question | How to handle error in destructor |
| A | Throwing |
| B | Terminate |
| C | Both a and b |
| D | None of the mentioned |
| Answer | B |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 112 |
| Question | In catch statement we have multiple parameters |
| A | Yes |
| B | No |
| C |  |
| D |  |
| Answer | B |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 113 |
| Question | ------kind of exceptions are in c++. |
| A | Handled |
| B | Static |
| C | Both a and b |
| D | Unhandled |
| Answer | A |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 114 |
| Question | Pick up the correct statement |
| A | To throw exception we have to use catch statement |
| B | Error occurring code is placed in try block |
| C | We can not have multiple throwing mechanism in c++ |
| D | Both and b |
| Answer | B |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 115 |
| Question | Can we used constructor for exception handling |
| A | Yes |
| B | No |
| C |  |
| D |  |
| Answer | A |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 116 |
| Question | Class template can be overloaded |
| A | True |
| B | False |
| C |  |
| D |  |
| Answer | B |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 117 |
| Question | \_\_\_\_\_\_is a generic class handler |
| A | Catch(---) |
| B | Catch(-,-) |
| C | Catch(…) |
| D | Catch(void) |
| Answer | C |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 118 |
| Question | If the exception is not handled then which standard library function get invoked |
| A | stop() |
| B | terminate() |
| C | Read() |
| D | Write() |
| Answer | B |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 119 |
| Question | Exception can be only built in type |
| A | True |
| B | False |
| C |  |
| D |  |
| Answer | A |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 120 |
| Question | What will be output of program  #include<iostream>  using namespace std;  template<class T>  T display(T x)  {  cout<< “using template x=”<<x<<“\n”;  }  int display(int x)  {  cout<<“Normal display x=”<<x <<“\n”;  }  int main()  {  display(2.3);  display(3);  diplay(1.1);  } |
| A | Normal display x=2.3  Using template x=3  Normal display x=1.1 |
| B | using template x=2.3  Normal display x=3  using template x=1.1 |
| C | using template x=3  Normal display x=2.3  using template x=1.1 |
| D | None of the above |
| Answer | B |
| Marks | 2 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 121 |
| Question | In nested try blocks, there is no need to specify catch handler for inner try block. Outer catch handler is sufficient for the program |
| A | True |
| B | False |
| C |  |
| D |  |
| Answer | B |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 122 |
| Question | Can we write try block within try block |
| A | True |
| B | False |
| C |  |
| D |  |
| Answer | A |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 123 |
| Question | Can we prevent a function from throwing any exceptions |
| A | Yes |
| B | No |
| C |  |
| D |  |
| Answer | A |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 124 |
| Question | What is return type of uncaught\_exception() is---- |
| A | Char\* |
| B | Double |
| C | Int |
| D | Bool |
| Answer | D |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 125 |
| Question | Can we write a throw statement inside catch statement |
| A | Yes |
| B | No |
| C |  |
| D |  |
| Answer | A |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 126 |
| Question | We can define our own exceptions in c++ |
| A | False |
| B | True |
| C |  |
| D |  |
| Answer | B |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 127 |
| Question | Stack unwinding deals with |
| A | Polymorphism |
| B | inheritance |
| C | Exception handling |
| D | Classes |
| Answer | C |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 128 |
| Question | What is STL |
| A | Standard Term Library |
| B | Standard Tree Library |
| C | Standard Template Library |
| D | None of the above mentioned |
| Answer | C |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 129 |
| Question | Pick up the correct statement   * Catch statement be placed immediately after try block * It can have multiple parameters * There must be multiple catch handler for a try block * Generic catch statement we can placed anywhere in program |
| A | i and ii |
| B | i and iii |
| C | i and iv |
| D | i , ii and iii |
| Answer | B |
| Marks | 2 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 130 |
| Question | Generic catch should be placed at |
| A | End of all statement |
| B | Before try |
| C | Before throw |
| D | Inside try |
| Answer | A |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 131 |
| Question | Irrespective of exception occurrence, catch handler will be always executed |
| A | Yes |
| B | No |
| C |  |
| D |  |
| Answer | B |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 132 |
| Question | From where does the template class can derived |
| A | Regular non-templated c++ class |
| B | Templated class |
| C | Both a and b |
| D | None of the above mentioned |
| Answer | C |
| Marks | 2 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 133 |
| Question | What is done by compiler for templates |
| A | Type-safe |
| B | Code elimination |
| C | Portability |
| D | All of the above mentioned |
| Answer | A |
| Marks | 2 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 134 |
| Question | Catch handler itself may detect and throw an exception |
| A | True |
| B | False |
| C |  |
| D |  |
| Answer | A |
| Marks | 1 |
| Unit | Iv |

|  |  |
| --- | --- |
| **Id** | 135 |
| Question | If the thrown exception will not be caught by any catch statement then it will be passed to next outer try/catch sequence for processing. |
| A | False |
| B | True |
| C |  |
| D |  |
| Answer | B |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 136 |
| Question | While specifying the exceptions, the type-list specifies the\_\_\_\_\_\_\_\_ that may be thrown. |
| A | How many exceptions |
| B | Type of exception |
| C | Both a and b |
| D | None of the above mentioned |
| Answer | B |
| Marks | 2 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 137 |
| Question | When an exception is rethrown ,it will not be caught by the\_\_\_\_\_\_\_\_\_\_or other catch in that group. |
| A | Same catch |
| B | Nested catch |
| C | Both a and b |
| D | None of the above mentioned |
| Answer | A |
| Marks | 2 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 138 |
| Question | Try block can throw any exception |
| A | True |
| B | False |
| C |  |
| D |  |
| Answer | B |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 139 |
| Question | Pick up the correct statement from the following   * Multiple catch statement are there in c++. * We have generic catch statement to handle all type of exception * Try block is used to throw and exception |
| A | i and iii |
| B | i |
| C | ii |
| D | i and ii only |
| Answer | D |
| Marks | 2 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 140 |
| Question | When an exception is not caught |
| A | Program is go in wait condition |
| B | Program is aborted |
| C | Program works fine way |
| D | None of the above mentioned |
| Answer | B |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 141 |
| Question | We can place two or more catch blocks together to catch and handle multiple types of exceptions thrown by a try blocks |
| A | True |
| B | False |
| C |  |
| D |  |
| Answer | A |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 142 |
| Question | It is also possible to make a catch statement to catch all types of exceptions using ellipses as its arguments |
| A | True |
| B | False |
| C |  |
| D |  |
| Answer | A |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 143 |
| Question | We can restrict a function to throw only a set of specified exceptions by adding a throw specification clause to the function definition. |
| A | True |
| B | False |
| C |  |
| D |  |
| Answer | A |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 144 |
| Question | We may also use non-type parameters such basic or derived data types as arguments template |
| A | True |
| B | False |
| C |  |
| D |  |
| Answer | A |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 145 |
| Question | Pick up the correct statement from the following related with overloading of template functions   * Call an ordinary function that has an exact match * Call a template function that could be created with an exact macth * Try normal overloading resolution to ordinary functions and call the one that matches |
| A | 1 and 2 only |
| B | 2 and 3 only |
| C | All of the above |
| D | None of the above mentioned |
| Answer | C |
| Marks | 2 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 146 |
| Question | What will be output of the a following program  #include<iostream>  using namespace std;  template <class T>  void display(T x)  {  cout<<“Template display:”<<x<< “\n”;  }  void display(int x)  {  cout<<“Explicit display:”<<x <<“\n”;  }  int main()  {  display(100);  display(12.34);  display(‘c’);  } |
| A | Template display:100  Template display:12.34  Template display: c |
| B | Explicit display:100  Template display:12.34  Template display: c |
| C | Explicit display:100  Template display:12.34  Explicit display: c |
| D | Template display:100  Template display:12.34  Template display: c |
| Answer | B |
| Marks | 2 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 147 |
| Question | What will be output of program  #include <iostream>  using namespace std;  int main()  {  cout <<"Start\n";  try {  cout <<"Inside try block\n";  throw 100;  cout << "This will not execute";  }  catch (int i) {  cout <<"Caught an exception -- value is: ";  cout <<i <<"\n";  }  cout <<"End";  return 0;  } |
| A | Start  Inside try block  Caught an exception -- value is: 100  End |
| B | Start  End |
| C | Start  Inside try block  End |
| D | None of the above mentioned |
| Answer | A |
| Marks | 2 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 148 |
| Question | What will be output of following program  #include <iostream>  using namespace std;  void Xhandler(int test)  {  try{  if(test) throw test;  else throw "Value is zero";  }  catch(int i) {  cout << "Caught Exception #: " <<i <<'\n';  }  catch(const char \*str) {  cout <<"Caught a string: ";  cout << str <<'\n';  }  }  int main()  {  cout <<"Start\n";  Xhandler(1);  Xhandler(2);  Xhandler(0);  Xhandler(3);  cout << "End";  return 0;  } |
| A | Start  Caught Exception #: 1  Caught Exception #: 2  Caught Exception #: 0  Caught Exception #: 3  End |
| B | Start  Caught Exception #: 1  Caught Exception #: 2  Caught a string: 0  Caught Exception #: 3  End |
| C | Start  Caught Exception #: 1  Caught Exception #: 2  Caught a string: Value is zero  Caught Exception #: 3  End |
| D | None of the mentioned |
| Answer | C |
| Marks | 2 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 149 |
| Question | What will be output of program  #include <iostream>  using namespace std;  void Xhandler(int test)  {  try{  if(test==0) throw test; // throw int  if(test==1) throw 'a'; // throw char  if(test==2) throw 123.23; // throw double  }  catch(int i) { // catch an int exception  cout <<"Caught an integer\n";  }  catch(...) { // catch all other exceptions  cout <<"Caught One!\n";  }  }  int main()  {  cout <<"Start\n";  Xhandler(0);  Xhandler(1);  Xhandler(2);  cout <<"End";  return 0;  } |
| A | Start  Caught One!  Caught One!  Caught One!  End |
| B |  |
| C | Start  Caught an integer  Caught One!  Caught One!  End |
| D | Start  Caught One!  Caught an integer  Caught One!  End |
| Answer | C |
| Marks | 2 |
| Unit | None of the above mentioned |

|  |  |
| --- | --- |
| **Id** | 150 |
| Question | What will be output of following program  #include <iostream>  using namespace std;  template <class Type1, class Type2> class myclass  {  Type1 i;  Type2 j;  public:  myclass(Type1 a, Type2 b) { i = a; j = b; }  void show() { cout <<i << ' ' <<j <<'\n'; }  };  int main()  {  myclass<int, double>ob1(10, 0.23);  myclass<char, char \*>ob2('X', "Templates add power.");  ob1.show(); // show int, double  ob2.show(); // show char, char \*  return 0;  } |
| A | 10 0.23  X Templates add power. |
| B | 0.23 10  X Template add power |
| C | 10 10  X template add power |
| D | Compilation error |
| Answer | A |
| Marks | 2 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 151 |
| Question | We can combine operator overloading with a class |
| A | True |
| B | False |
| C |  |
| D |  |
| Answer | A |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 152 |
| Question | If you overload a generic function, that overloaded function overrides (or "hides") the generic function relative to that specific version. |
| A | True |
| B | False |
| C |  |
| D |  |
| Answer | A |
| Marks | 1 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 153 |
| Question | What will be output of following programming  #include <iostream>  using namespace std;  template <class T>  T GetMax (T a, T b) {  T result;  result = (a>b)? a : b;  return (result);  }  int main () {  int i=5, j=6, k;  long l=10, m=5, n;  k=GetMax<int>(i,j);  n=GetMax<long>(l,m);  cout <<k << endl;  cout <<n <<endl;  return 0;  } |
| A | 6  10 |
| B | 5  5 |
| C | 10  10 |
| D | Compilation error |
| Answer | A |
| Marks | 2 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 154 |
| Question | What will be output of following program  #include <iostream>  using namespace std;  template <class T>  class mypair {  T a, b;  public:  mypair (T first, T second)  {a=first; b=second;}  T getmax ();  };  template <class T>  T mypair<T>::getmax ()  {  T retval;  retval = a>b? a : b;  return retval;  }  int main () {  mypair <int>myobject (100, 75);  cout << myobject.getmax();  return 0;  } |
| A | 75 |
| B | 100 |
| C | 75  100 |
| D | Compilation error |
| Answer | B |
| Marks | 2 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 155 |
| Question | What will be output of following program  #include <iostream>  #include <exception>  using namespace std;  class myexception: public exception  {  virtual const char\* what() const throw()  {  return "My exception happened";  }  } myex;  int main () {  try  {  throw myex;  }  catch (exception&e)  {  cout << e.what() <<endl;  }  return 0;  } |
| A | Exception happened |
| B | My exception happened. |
| C | Run Time error |
| D | Compilation error |
| Answer | B |
| Marks | 2 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 156 |
| Question | Pick up the correct statement from following  1.Exception handling is not supported c++  2.Template support generic programming in c++  3.overloading of function template is possible in c++  4.generic catch template can handle all types of exceptions |
| A | 2 and 3 only |
| B | 3 and 4 only |
| C | 1, 2 and 3 only |
| D | 2, 3 and 4 only |
| Answer | D |
| Marks | 2 |
| Unit | IV |

|  |  |
| --- | --- |
| **Id** | 1 |
| Question | Polymorphism is supported by the c++ by using following ways |
| A | function overloading |
| B | operator overloading |
| C | virtual functions |
| D | all of the above |
| Answer | D |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 2 |
| Question | Compile time polymorphism is supported by |
| A | function overloading |
| B | virtual function |
| C | operator overloading |
| D | both a&c |
| Answer | D |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 3 |
| Question | Run time polymorphism is supported by |
| A | function overloading |
| B | operator overloading |
| C | virtual function |
| D | both a and b |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 4 |
| Question | Selecting the appropriate overloaded function by the compiler is known as |
| A | late binding |
| B | early binding |
| C | both a and b |
| D | none of the above |
| Answer | B |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 5 |
| Question | object to function binding is done at compile time then is it known as |
| A | early binding |
| B | compile time binding |
| C | none of the above |
| D | both a and b |
| Answer | D |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 6 |
| Question | Run time polymorphism is done by using |
| A | function overloading |
| B | operator overloading |
| C | virtual function |
| D | none of the above |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 7 |
| Question | Operator overloading is ---. |
| A | run time polymorphism |
| B | compile time polymorphism |
| C | none of the above |
| D | both a and b |
| Answer | B |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 8 |
| Question | Which of the following operator cannot be overloaded |
| A | scope resolution operator(::) |
| B | Size of operator (sizeof[]) |
| C | Conditional operator(?:) |
| D | All of the above |
| Answer | D |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 9 |
| Question | Which of the operator cannot be overloaded |
| A | >= |
| B | & |
| C | <= |
| D | :: |
| Answer | D |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 10 |
| Question | While performing operator overloading which function/keyword we have to use |
| A | Function |
| B | Operator |
| C | Op |
| D | none of the above |
| Answer | B |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 11 |
| Question | Which of the statement is not true about operator overloading |
| A | we can overload only existing operator |
| B | basic meaning cannot be changed |
| C | binary operator should have return type |
| D | All of the above |
| Answer | D |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 12 |
| Question | Pick up the correct statement related with operator overloading |
| A | we can overload a class access operator |
| B | we can change the meaning of basic operator |
| C | binary operator should have a return type |
| D | both a and b |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 13 |
| Question | We are overloading a unary operator without friend function how many argument we have to pass |
| A | 1 |
| B | 2 |
| C | 0 |
| D | none of the above |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 14 |
| Question | Suppose we are overloading a binary operator with friend function, how many parameter of argument we have to pass |
| A | 1 |
| B | 2 |
| C | 3 |
| D | none of ths above |
| Answer | B |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 15 |
| Question | we are overloading a binary operator without friend function how many argument we have to pass |
| A | 1 |
| B | 2 |
| C | 0 |
| D | none of the abve |
| Answer | A |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 16 |
| Question | What is polymorphism |
| A | it is ability to take many forms |
| B | it is instance of class |
| C | one class acquire the properties of another class |
| D | All of the above |
| Answer | A |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 17 |
| Question | What is true about the operator overloading |
| A | with friend function we need to pass two arguments for binary operator |
| B | with friend function we need to pass one arguments for unary operator |
| C | both a and b |
| D | none of the above |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 18 |
| Question | \_\_\_\_\_\_allows you to give special meaning to some operator when there are operands associated with it. |
| A | function overloading |
| B | virtual function |
| C | operator overloading |
| D | none of the above |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 19 |
| Question | Converting from small to larger data type is known as \_\_\_\_. |
| A | promotion |
| B | operator |
| C | polymorphism |
| D | none of the above |
| Answer | A |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 20 |
| Question | what are the types of type conversion |
| A | implicit |
| B | explicit |
| C | both a and b |
| D | none of the above |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 21 |
| Question | Reusability is supported by following feature |
| A | polymorphisms |
| B | message passing |
| C | inheritance |
| D | operator overloading |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 22 |
| Question | Deriving a new class from a base class is known as \_\_. |
| A | polymorphisms |
| B | inheritance |
| C | message passing |
| D | operator overloading |
| Answer | B |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 23 |
| Question | Base class is also known as\_\_\_. |
| A | super class |
| B | parent class |
| C | both a and b |
| D | none of the above |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 24 |
| Question | Child class is also known as |
| A | sub class |
| B | derived class |
| C | both a and b |
| D | known class |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 25 |
| Question | Derived class\_\_\_ cannot access from base class |
| A | constructor |
| B | destructors |
| C | copy constructor |
| D | all of the above |
| Answer | D |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 26 |
| Question | we can derive a new class from a derived class |
| A | true |
| B | false |
| C |  |
| D |  |
| Answer | A |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 27 |
| Question | How many parameter does a conversion operator take? |
| A | 0 |
| B | 2 |
| C | 3 |
| D | as many as possible |
| Answer | A |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 28 |
| Question | \_\_is used to define pure virtual function? |
| A | & |
| B | =0 |
| C | @ |
| D | \* |
| Answer | B |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 29 |
| Question | Which is also known as abstract class? |
| A | virtual function |
| B | pure virtual function |
| C | derived class function |
| D | base class function |
| Answer | B |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 30 |
| Question | pick the correct option |
| A | We can make the instance of the abstract class |
| B | We can not make the instance of the abstract class |
| C | both a and b |
| D | none of the above |
| Answer | B |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 31 |
| Question | How many access specifiers are there in c++? |
| A | 2 |
| B | 3 |
| C | 5 |
| D | 4 |
| Answer | B |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 32 |
| Question | Where we have to use an abstract class? |
| A | in base class only |
| B | in derived class only |
| C | both and b |
| D | None of the above |
| Answer | A |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 33 |
| Question | For what we can apply access |
| A | function |
| B | data member |
| C | none of the above specifier |
| D | both a and c |
| Answer | D |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 34 |
| Question | What is default access specifier for class? |
| A | public |
| B | protected |
| C | private |
| D | none of the above |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 35 |
| Question | We have to define a constructor for the derived class must be required\_\_\_\_. |
| A | if base class constructor does not require arguments |
| B | if base/parent class constructor required arguments |
| C | no need |
| D | always |
| Answer | B |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 36 |
| Question | Use of the friend function is\_\_\_. |
| A | the class allowing access to another class |
| B | the private section of a class |
| C | the public section of the class |
| D | all of the above |
| Answer | D |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 37 |
| Question | If an attribute is private define then which method can have access to it |
| A | only static function |
| B | only functions of that class |
| C | only method in that package |
| D | none of the above |
| Answer | B |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 38 |
| Question | What is syntax of deriving a new class from base class is\_\_\_. |
| A | class name, new class name |
| B | new class name, base class name |
| C | class name: access specifier class name |
| D | none of the above |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 39 |
| Question | Which constructor will initialize the base class data member |
| A | base class |
| B | derived class |
| C | derived derived class |
| D | none of these |
| Answer | A |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 40 |
| Question | Inheritance can be done using :: symbol |
| A | True |
| B | False |
| C |  |
| D |  |
| Answer | B |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 41 |
| Question | When we derived a new class using more than one class then type of inheritance is known as\_\_\_\_. |
| A | multiple inheritance |
| B | single inheritance |
| C | hybrid inheritance |
| D | multilevel inheritance |
| Answer | A |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 42 |
| Question | When class B is derive from A , and class C is derived from B, this kind of inheritance is known as \_\_\_\_\_\_. |
| A | multiple inheritance |
| B | single inheritance |
| C | hybrid inheritance |
| D | multilevel inheritance |
| Answer | D |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 43 |
| Question | The base class will provide you\_\_\_\_\_\_\_\_\_\_. |
| A | specific objects than the derived class |
| B | more generalized version of derived class |
| C | empty template of base class |
| D | all of the above |
| Answer | B |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 44 |
| Question | Pick up the correct statement form following  i)we have to use abstract keyword to define the abstract class  ii)inheritance allows multilevel class hierarchies more than two levels also  iii)reusability is supported by derivation  iv) we can change the meaning of operator during operator overloading |
| A | i only |
| B | ii only |
| C | i and ii only |
| D | ii and iii only |
| Answer | D |
| Marks | 2 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 45 |
| Question | What does derived class does not inherit from the base class  i)constructor  ii)destructor  iii)operator=()  iv)friends |
| A | i and ii |
| B | ii and iii |
| C | only i, iii and iv |
| D | all of the above |
| Answer | D |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 46 |
| Question | Choose the correct the statement from following  i)inheritance supported in terms of single, multiple , multilevel ,hybrid inheritance  ii) polymorphisms is supported by function overloading , operator overloading and virtual function  iii) abstraction is not supported by c++ |
| A | i is correct |
| B | only ii is correct |
| C | i and ii is correct |
| D | none of the above |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 47 |
| Question | \_\_\_\_\_types of classes in c++. |
| A | 1 |
| B | 2 |
| C | 3 |
| D | 4 |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 48 |
| Question | :\_\_\_\_\_\_\_is used to define a pure virtual function. |
| A | $ |
| B | ^ |
| C | =0 |
| D | # |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 49 |
| Question | Pick up the correct statement |
| A | a base class may have more than one class |
| B | derived class can be derived from more than one class |
| C | both a and b |
| D | none of the above |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 50 |
| Question | class A: public B, public C is a type of inheritance |
| A | Single |
| B | Multiple |
| C | Multilevel |
| D | Hybrid |
| Answer | B |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 51 |
| Question | When we have to use the mutable keyword |
| A | data member to change within a const member function |
| B | not allow the data member to change within a const member function |
| C | it will copy the values of the variable |
| D | none of the above mentioned |
| Answer | A |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 52 |
| Question | choose the correct statement |
| A | destructor cannot be inherited |
| B | private member not inherited to derived class |
| C | constructor cannot inherited |
| D | a and c |
| Answer | D |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 53 |
| Question | Use of function or operator to act different ways on different data type is called as |
| A | derivation |
| B | inheritance |
| C | polymorphisms |
| D | none of the above |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 54 |
| Question | Choose the correct statement |
| A | Constructor has a return type |
| B | constructor always define in public scope |
| C | constructor has same name that of class name |
| D | b and c |
| Answer | D |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 55 |
| Question | overloading of a prefix increment operator by means of a member function takes\_\_\_\_\_\_\_\_\_. |
| A | one argument |
| B | two argument |
| C | no argument |
| D | none of the above |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 56 |
| Question | Pick up the correct statement  i)abstract type of class should contain at least one virtual function  ii)we can create an object of abstract class  iii)abstract class is used to provide an interface to subclasses  iv)abstract class can also have normal function |
| A | i and ii only |
| B | i , ii, and iii only |
| C | i ,iii and iv |
| D | all of the above |
| Answer | D |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 57 |
| Question | What is function overloading |
| A | we have to use same function name but different parameter |
| B | different function name but same parameter |
| C | both a and b |
| D | none of the above |
| Answer | A |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 58 |
| Question | Virtual base class is used to \_\_\_\_. |
| A | to perform operator overloading |
| B | to perform function overloading |
| C | to remove ambiguity in multiple inheritance |
| D | all of the above |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 59 |
| Question | Pick up the correct statement |
| A | protected member from base class can be accessed by own class and its all subclasses |
| B | protected member are not inherited by any other class |
| C | Protected member are combination of public and private access member |
| D | all of the above |
| Answer | D |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 60 |
| Question | pick up the correct statement |
| A | base class and derived class can have their own constructor |
| B | base class and derived class can have their own destructor |
| C | neither a or neither b |
| D | both a and b |
| Answer | D |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 61 |
| Question | Make a correct sequence of a statement  i)destructor of derived class is called  ii)destructor of base class is called  iii)constructor of derived class is called  iv)constructor of base class is called |
| A | i,ii,iv,iii |
| B | iv,iii,ii,i |
| C | iv,iii,i,ii |
| D | i,ii,iii,iv |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 62 |
| Question | Operator overloading is |
| A | making C++ operators work with objects. |
| B | giving C++ operators more than they can handle. |
| C | giving new meanings to existing C++ operators. |
| D | Both A and C |
| Answer | D |
| Marks | 2 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 63 |
| Question | Assume a class C with objects obj1, obj2, and obj3. For the statement obj3 = obj1 - obj2 to work correctly, the overloaded - operator must |
| A | take two arguments. |
| B | return a value. |
| C | use the object of which it is a member as an operand. |
| D | Both B and C |
| Answer | D |
| Marks | II |
| Unit | 2 |

|  |  |
| --- | --- |
| **Id** | 64 |
| Question | When you overload an arithmetic assignment operator, the result |
| A | goes in the object to the right of the operator. |
| B | goes in the object to the left of the operator. |
| C | goes in the object of which the operator is a member. |
| D | Both B and C |
| Answer | D |
| Marks | 2 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 65 |
| Question | To convert from a user-defined class to a basic type, you would most likely use |
| A | a built-in conversion operator. |
| B | a one-argument constructor. |
| C | an overloaded = operator. |
| D | a conversion operator that’s a member of the class. |
| Answer | D |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 66 |
| Question | An overloaded operator always requires one less argument than its number  of operands. |
| A | TRUE |
| B | FALSE |
| C |  |
| D |  |
| Answer | A |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 67 |
| Question | The compiler won’t object if you overload the \* operator to perform division. |
| A | TRUE |
| B | FALSE |
| C |  |
| D |  |
| Answer | A |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 68 |
| Question | Inheritance is a way to |
| A | make general classes into more specific classes. |
| B | pass arguments to objects of classes. |
| C | add features to existing classes without rewriting them. |
| D | A and C |
| Answer | D |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 69 |
| Question | Advantages of inheritance include |
| A | providing a useful conceptual framework. |
| B | facilitating class libraries. |
| C | avoiding the rewriting of code. |
| D | All of the above |
| Answer | D |
| Marks | 2 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 70 |
| Question | Adding a derived class to a base class requires fundamental changes to the base class. |
| A | TRUE |
| B | FALSE |
| C |  |
| D |  |
| Answer | D |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 71 |
| Question | To be accessed from a member function of the derived class, data or functions in the base class must be |
| A | public |
| B | private |
| C | protected |
| D | static |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 72 |
| Question | If a base class contains a member function basefunc(), and a derived class does not contain a function with this name, can an object of the derived class access basefunc()? |
| A | YES |
| B | NO |
| C |  |
| D |  |
| Answer | A |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 73 |
| Question | If no constructors are specified for a derived class, objects of the derived class will use the constructors in the base class. |
| A | TRUE |
| B | FALSE |
| C |  |
| D |  |
| Answer | A |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 74 |
| Question | The scope-resolution operator usually |
| A | specifies a particular class. |
| B | tells what base class a class is derived from. |
| C | resolves ambiguities. |
| D | A and C |
| Answer | D |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 75 |
| Question | Assume a class **Derv** that is privately derived from class **Base**. An object of class **Derv** located in **main()** can access |
| A | public members of **Derv.** |
| B | protected members of **Derv**. |
| C | private members of **Derv**. |
| D | public members of **Base**. |
| Answer | A |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 76 |
| Question | True or False: A class **D**can be derived from a class **C**, which is derived from a class **B**, which is derived from a class **A**. |
| A | TRUE |
| B | FALSE |
| C |  |
| D |  |
| Answer | A |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 77 |
| Question | It is illegal to make objects of one class members of another class. |
| A | TRUE |
| B | FALSE |
| C |  |
| D |  |
| Answer | B |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 78 |
| Question | A class hierarchy |
| A | shows the same relationships as an organization chart. |
| B | describes “has a” relationships. |
| C | describes “is a kind of” relationships. |
| D | shows the same relationships as a family tree. |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 79 |
| Question | What is the output of the program?  #include <iostream>  #include <string>  using namespace std;  class Department {  public:  string dept;  Department(string d):dept(d) { }  void getDeptName() { cout <<dept; }  };  class Student : private Department {  public:  string name;  Student(string n = "Not entered", string d = "ATDC") :  name(n), Department(d) { }  using Department::getDeptName;  };  int main() {  Student s("CSE");  s.getDeptName();  return 0;  } |
| A | CSE |
| B | ATDC |
| C | Not entered |
| D | Compilation Error |
| Answer | B |
| Marks | 2 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 80 |
| Question | Identify the lines on which the compiler will report an error.  #include <iostream>// ---1  using namespace std; // ---2  class Base { // ---3  int var\_; // ---4  public: // ---5  Base():var\_(0){} // ---6  }; // ---7  class Derived: public Base { public: // ---8  int varD\_; // ---9  void print () { cout <<var\_; } // ---10  }; // ---11  int main() { // ---12  Derived d; // ---13  d.var\_ = 1; // ---14  d.varD\_ = 1; // ---15  cout <<d.var\_ <<""<<d.varD\_; // ---16  return 0; // ---17  } // ---18 |
| A | 6, 10, 14, 15 |
| B | 6, 15 |
| C | 6, 14, 16 |
| D | 10, 14, 16 |
| Answer | D |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 81 |
| Question | #include <iostream>  using namespace std;  class Base { public:  int var\_;  void func(int){}  };  class Derived: public Base { public:  int varD\_;  void func(int){}  };  int main() {  Derived d;  d.func(1);  return 0;  }  Which of the following function will be invoked by d.func(1)? |
| A | Base::func(int) |
| B | Derived::func(int) |
| C | Compilation Error |
| D | None of the above |
| Answer | B |
| Marks | 2 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 82 |
| Question | What is the output of the following program?  #include<iostream>  #include<string>  using namespace std;  class Base {  public:  void func\_f1(int i) { cout <<"In base func\_f1 "; }  void func\_f2(int i) { cout <<"In base func\_f2 "; }  };  class Derived: public Base {  public:  void func\_f1(int i ) { cout <<"In derived func\_f1 "; }  void func\_f1(string s) { cout <<"func\_f1 string "; }  void func\_f3(int i) { cout <<"In derived func\_f3 "; }  };  int main() {  Base b;  Derived d;  d.func\_f1(3);  d.func\_f1("Blue");  d.func\_f3(3);  d.func\_f2(3);  return 0;  } |
| A | Compilation Error: Cannot add new parameters to func\_f1 |
| B | In derived func\_f1 func\_f1 string In derived func\_f3 In base func\_f2 |
| C | In base func\_f2 func\_f1 string In derived func\_f3 In derived func\_f1 |
| D | Compilation Error: Cannot define func\_f3 containing same parameter type as func\_f1 |
| Answer | B |
| Marks | 2 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 83 |
| Question | What is the output of the following program? {Assume size of int as 4}  #include<iostream>  using namespace std;  class base {  int data;  };  class derived1: public base { };  class derived2: public derived1 { };  int main() {  cout <<sizeof(derived2);  return 0;  } |
| A | 4 |
| B | 8 |
| C | 12 |
| D | 16 |
| Answer | A |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 84 |
| Question | What will be the output of the following program?  #include <iostream>  using namespace std;  class B{ public: int base;  B() {}  ~B() {}  };  class D: public B { public: int derived;  D() {}  ~D() {}  };  int main() {  D d1;  B b1;  cout <<&b1.base <<"";  cout <<&d1.base;  return 0;  } |
| A | 0x28fef8 0x28fef8 |
| B | 0x28fef8 0x28fefc |
| C | Compilation Error |
| D | None of the above |
| Answer | B |
| Marks | 2 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 85 |
| Question | What will be the output of the following program?  #include<iostream>  using namespace std;  class Base { public:  Base() { cout <<"Base Ctor"<<endl; }  ~Base() { cout <<"Base Dtor"<<endl; }  };  class Derived: public Base { public:  Derived() { cout <<"Derived Ctor"<<endl; }  ~Derived() { cout <<"Derived Dtor"<<endl; }  };  int main() {  Derived d1;  {  Base b1;  }  return 0;  } |
| A | Base Ctor  Derived Ctor  Base Ctor  Base Dtor  Base Dtor  Derived Dtor |
| B | Derived Ctor  Base Ctor  Base Ctor  Base Dtor  Derived Dtor  Base Dtor |
| C | Derived Ctor  Base Ctor  Base Dtor  Derived Dtor |
| D | Base Ctor  Derived Ctor  Base Ctor  Base Dtor  Derived Dtor  Base Dtor |
| Answer | D |
| Marks | 2 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 86 |
| Question | What will be the output of the program?  #include <iostream>  using namespace std;  class F1 {  public:  F1() { cout <<"F1 ctor "; }  ~F1() { cout <<"F1 dtor "; }  };  class F2 : public F1 {  public:  F2() { cout <<"F2 ctor "; }  ~F2() { cout <<"F2 dtor "; }  };  class F3 : public F1 {  const F2 &f2;  public:  F3() : f2(\*new F2) { cout <<"F3 ctor "; }  ~F3() { cout <<"F3 dtor "; }  };  int main() {  F3 f3;  return 0;  } |
| A | F1 ctor F2 ctor F3 ctor F3 dtor F2 dtor F1 dtor |
| B | F1 ctor F1 ctor F2 ctor F3 ctor F3 dtor F1 dtor |
| C | F1 ctor F3 ctor F3 dtor F1 dtor |
| D | F1 ctor F1 ctor F2 ctor F3 ctor F3 dtor F2 dtor F1 dtor F1 dtor |
| Answer | B |
| Marks | 2 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 87 |
| Question | What will be the output of the program?  #include <iostream>  using namespace std;  class Room {  int number;  public:  Room(int num = 0): number(num) { }  void dimension() { cout <<number <<"Rooms "; }  };  class Building {  public:  Building() : ro(100) { }  void Build() { ro.dimension(); }  private:  Room ro;  };  int main() {  Building B;  B.Build();  return 0;  } |
| A | 0 Rooms |
| B | 100 Rooms |
| C | Compilation Error: ro is private |
| D | None of the above |
| Answer | B |
| Marks | 2 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 88 |
| Question | What will be the output of the program?  #include<iostream>  using namespace std;  class Shape {  public:  int x, y;  Shape(int a = 0, int b = 0): x(a), y(b) {}  void draw()  { cout <<x <<""<<y <<""; }  };  class Rectangle : public Shape {  public:  int w, h;  Rectangle(int a = 5, int b = 6): w(a), h(b), Shape(7, 8) {}  void draw()  { Shape::draw(); cout <<w <<""<<h ; }  };  int main() {  Rectangle \*r = new Rectangle(1,2);  r->draw();  return 0;  } |
| A | 0 0 1 2 |
| B | 7 8 1 2 |
| C | 7 8 5 6 |
| D | 0 0 5 6 |
| Answer | B |
| Marks | 2 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 89 |
| Question | You cannot change the precedence and associativity of an operator by overloading. |
| A | TRUE |
| B | FALSE |
| C |  |
| D |  |
| Answer | A |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 90 |
| Question | When deriving a class from with **protected** inheritance, **public** members of the base class become\_\_\_\_\_\_\_\_\_\_\_ members of the derived class, and **protected** members of the  base class become \_\_\_\_\_\_\_\_\_\_\_\_\_ members of the derived class. |
| A | protected, protected. |
| B | public, private |
| C | private, private |
| D | Private, protected |
| Answer | A |
| Marks | 2 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 91 |
| Question | When deriving a class with **public** inheritance, **public** members of the base class become \_\_\_\_\_\_\_\_\_\_ members of the derived class, and **protected** members of the base class become \_\_\_\_\_\_\_\_\_\_\_members of the derived class. |
| A | private, private |
| B | public, protected. |
| C | protected, protected. |
| D | private, protected |
| Answer | B |
| Marks | 2 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 92 |
| Question | C++ provides for \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, which allows a derived class to inherit from many base classes, even if the base classes are unrelated. |
| A | Multilevel inheritance |
| B | Single level inheritance |
| C | multiple inheritance |
| D | Hierarchical inheritance |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 93 |
| Question | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_is a form of software reuse in which new classes absorb the data and behaviors of existing classes and embellish these classes with new capabilities. |
| A | Data hiding |
| B | Inheritance |
| C | Abstraction |
| D | encapsulation |
| Answer | B |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 94 |
| Question | We can create the object of abstract class |
| A | True |
| B | False |
| C |  |
| D |  |
| Answer | B |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 95 |
| Question | All **virtual** functions in an abstract base class must be declared as **pure virtual** functions. |
| A | TRUE |
| B | FALSE |
| C |  |
| D |  |
| Answer | B |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 96 |
| Question | A class is made abstract by declaring that class virtual. |
| A | TRUE |
| B | FALSE |
| C |  |
| D |  |
| Answer | B |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 97 |
| Question | Polymorphic programming can eliminate the need for **switch** logic. |
| A | TRUE |
| B | FALSE |
| C |  |
| D |  |
| Answer | A |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 98 |
| Question | Suppose **a** and **b** are integer variables and we form the sum **a + b**. Now suppose **c** and  **D** are floating-point variables and we form the sum **c + d.** The two **+**operators here are  clearly being used for different purposes. This is an example of \_\_\_\_\_\_\_\_\_\_\_\_ |
| A | Operator Overloading |
| B | Inheritance |
| C | Function Overloading |
| D | Constructor |
| Answer | A |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 99 |
| Question | The operators that cannot be overloaded is |
| A | \* |
| B | - |
| C | :: |
| D | () |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 100 |
| Question | The operators that cannot be overloaded is |
| A | \* |
| B | ?: |
| C | >> |
| D | << |
| Answer | B |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 101 |
| Question | Which of the following operator(s) can not be overloaded |
| A | .\* |
| B | :: |
| C | ?: |
| D | All of the above |
| Answer | D |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 102 |
| Question | Which of the following is true about this pointer? |
| A | It is passed as a hidden argument to all function calls |
| B | It is passed as a hidden argument to all non-static function calls |
| C | It is passed as a hidden argument to all static functions |
| D | None of the above |
| Answer | B |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 103 |
| Question | Predict the output of following C++ program.  #include<iostream>  using namespace std;    class Test  {  private:  int x;  public:  Test(int x = 0) { this->x = x; }  void change(Test \*t) { this = t; }  void print() { cout <<"x = "<<x <<endl; }  };    int main()  {  Test obj(5);  Test \*ptr = new Test (10);  obj.change(ptr);  obj.print();  return 0;  } |
| A | x = 5 |
| B | x = 10 |
| C | Compiler Error |
| D | Runtime Error |
| Answer | C |
| Marks | 2 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 104 |
| Question | Which of the followings is/are automatically added to every class, if we do not write our own. |
| A | Copy Constructor |
| B | Assignment Operator |
| C | A constructor without any parameter |
| D | All of the above |
| Answer | D |
| Marks | 2 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 105 |
| Question | What is the output of following program?  #include<iostream>  using namespace std;  class Point {  Point() { cout <<"Constructor called"; }  };    int main()  {  Point t1;  return 0;  } |
| A | Compiler Error |
| B | Runtime Error |
| C | Constructor called |
| D | Segmentation Fault |
| Answer | A |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 106 |
| Question | What will be the output of following program?  #include <iostream>  using namespace std;    class Test  {  public:  Test() { cout <<"Hello from Test() "; }  } a;    int main()  {  cout <<"Main Started ";  return 0;  } |
| A | Main Started |
| B | Main Started Hello from Test() |
| C | Hello from Test() Main Started |
| D | Compiler Error: Global objects are not allowed |
| Answer | C |
| Marks | 2 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 107 |
| Question | Which of the following operators are overloaded by default by the compiler?  1) Comparison Operator ( == )  2) Assignment Operator ( = ) |
| A | Both 1 and 2 |
| B | Only 1 |
| C | Only 2 |
| D | None of the two |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 108 |
| Question | A normal C++ operator that acts in a special way on newly defined data types is called \_\_\_\_\_\_\_ |
| A | Encapsulated |
| B | Overloaded |
| C | Classified |
| D | Inherited |
| Answer | B |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 109 |
| Question | The correct function name for overloading the addition + operator is \_\_ |
| A | Operator \_+ |
| B | Operator :+ |
| C | Operator (+) |
| D | Operator + |
| Answer | D |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 110 |
| Question | Which of the following operators cannot be overloaded? |
| A | → operator |
| B | . operator |
| C | [ ] operator |
| D | &operator |
| Answer | B |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 111 |
| Question | Which of the following operators cannot be overloaded? |
| A | + |
| B | - |
| C | [ ] |
| D | :: |
| Answer | D |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 112 |
| Question | Pick the incorrect statement from the following |
| A | The overloaded operators follow the syntax rules of original operator. |
| B | Only existing operators can be overloaded |
| C | Overloaded operator must have at least one operand of its class type |
| D | Overloaded operators can change the meaning of the original operator |
| Answer | D |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 113 |
| Question | For operators to be overloaded as non static member functions: |
| A | Both binary and unary operators take one argument. |
| B | Binary operators can have one argument and unary operators can not have any |
| C | Neither binary nor unary operators can have arguments |
| D | Binary operators can have two arguments and unary operators can have one |
| Answer | B |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 114 |
| Question | Which of the following is an operator function? |
| A | Member overloading |
| B | Function overloading |
| C | Operator overloading |
| D | None of these |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 115 |
| Question | Operator overloading means \_\_\_\_\_\_\_ |
| A | Giving new meaning to existing operator without changing its original  Meaning |
| B | Making C++ operators to work with objects |
| C | Making new types of operator |
| D | Both a and b |
| Answer | D |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 116 |
| Question | For overloading =+ implicitly \_\_\_\_\_\_\_ |
| A | + and = operators need to be overloaded implicitly |
| B | Only + operator need to be overloaded implicitly |
| C | Only = operator need to be overloaded implicitly |
| D | The += operator cannot be overloaded implicitly |
| Answer | D |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 117 |
| Question | Overloading a postfix increment operator by means of a member function takes------- |
| **A** | No argument |
| B | One argument |
| C | Two arguments |
| D | Three Arguments |
| Answer | A |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 118 |
| Question | If you overload only prefix operator ++ then the postfix ++ operator is \_\_\_\_\_\_ |
| A | Does not work |
| B | Works arbitrarily |
| C | Works naturally |
| D | Works as if prefix ++ operator |
| Answer | D |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 119 |
| Question | When compiler decides binding of an overloaded member then it is called\_\_\_\_\_\_\_\_ |
| A | Static binding |
| B | Dynamic binding |
| C | Local binding |
| D | None of these |
| Answer | A |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 120 |
| Question | One can redefine the working of \_\_\_\_\_\_\_ to work with objects. |
| A | Preprocessor directives |
| B | White space characters |
| C | Standard operators |
| D | None of these |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 121 |
| Question | Choose the correct option:  I. When you overload <<operator the >>operator automatically gets  overloaded  II. You can overload unary operator to work with binary operator |
| A | Only I is true |
| B | Only II is true |
| C | Both I and II are true |
| D | Neither I nor II are true |
| Answer | D |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 122 |
| Question | Choose the correct option  I.If you do not want to make use of operator overloading, you can achieve that effect using user defined function  II. The sizeof operator can be overloaded |
| A | Only I is true |
| B | Only II is true |
| C | Both I and II are true |
| D | Neither I nor II are true |
| Answer | A |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 123 |
| Question | The array subscript operator [] when overloaded cannot \_\_\_\_\_\_ |
| A | Take user defined objects are operands |
| B | Take float as an operand |
| C | Take multiple values inside (for example: [5,7] ) |
| D | None of these |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 124 |
| Question | The prototype of overloaded cast operator functions do not \_\_\_\_\_\_\_ |
| A | specify the type they convert to |
| B | specify the return type |
| C | need to be defined inside the class whose objects are being converted |
| D | none of these |
| Answer | B |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 125 |
| Question | Which of the following operators cannot be overloaded ? |
| A | += |
| B | << |
| C | ?: |
| D | FUNCTION CALL() |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 126 |
| Question | Which of the following operators cannot be overloaded ? |
| A | :: |
| B | Sizeof |
| C | Conditional operator ?: |
| D | All of these |
| Answer | D |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 127 |
| Question | The overloading the function operator\_\_\_\_\_\_\_\_. |
| A | requires class with overloaded operators |
| B | makes use of parameterized constructor |
| C | allows to create objects that are syntactically like functions |
| D | none of these |
| Answer | A |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 128 |
| Question | Choose the incorrect statement from the following. |
| A | Constructors can be overloaded. |
| B | Only existing operators must be overloaded |
| C | the overloaded operators must follow the syntax rules of the original operator |
| D | The overloaded operators must have atleast one operand of its class type |
| Answer | B |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 129 |
| Question | Overloading without explicit arguments to an operator function is called\_\_\_\_\_\_. |
| A | unary operator |
| B | binary operator |
| C | nested class |
| D | none of these |
| Answer | A |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 130 |
| Question | In binary overloaded function which are overloaded through friend function take\_\_\_\_\_\_\_ |
| A | three explicit arguments |
| B | two explicit arguments |
| C | one explicit argument |
| D | no argument |
| Answer | B |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 131 |
| Question | In binary overloaded function which are overloaded through member function take\_\_\_\_\_\_\_\_\_\_ |
| A | three explicit arguments |
| B | two explicit arguments |
| C | one explicit argument |
| D | no argument |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 132 |
| Question | The unary operators are overloaded by member function then it takes \_\_\_\_\_\_ |
| A | three explicit arguments |
| B | two explicit arguments |
| C | one explicit argument |
| D | no argument |
| Answer | D |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 133 |
| Question | Choose the correct choice.  I. All the operators in C++ can be overloaded.  II. We can change the basic meaning of operator while overloading it. |
| A | Only I is true |
| B | Only II is true |
| C | Both I and II are true |
| D | Neither I nor II are true |
| Answer | D |
| Marks | 1 |
| Unit | 2 |

|  |  |
| --- | --- |
| **Id** | 134 |
| Question | Which of the following operator can be overloaded through friend function ? |
| A | :: |
| B | + |
| C | = |
| D | -> |
| Answer | B |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 135 |
| Question | The name of the operator function that overloads the / symbol is\_\_\_\_\_\_\_\_. |
| A | operator /() |
| B | /op() |
| C | / operator() |
| D | op/() |
| Answer | A |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 136 |
| Question | In binary operator overloaded operator function the second operand should be\_\_\_\_\_\_. |
| A | passed by value |
| B | Implicit |
| C | passed by reference |
| D | none of these |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 137 |
| Question | Function overloading is run time polymorphisms |
| A | True |
| B | False |
| C |  |
| D |  |
| Answer | B |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 138 |
| Question | Following overloaded operator cannot be inherited by derived class\_\_\_\_\_\_\_. |
| A | > |
| B | = |
| C | \* |
| D | / |
| Answer | B |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 139 |
| Question | Choose the correct choice. |
| A | The conditional operator can be overloaded |
| B | While overloading using the friend function the binary operator requires one argument |
| C | Operator precedence cannot be changed |
| D | None of these |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 140 |
| Question | Which of the following operator can be overloaded through friend function ? |
| A | () |
| B | [] |
| C | -> |
| D | \* |
| Answer | D |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 141 |
| Question | When we overload we want to\_\_\_\_\_\_. |
| A | compare and copy object |
| B | assign one object to another |
| C | compare two objects |
| D | test for equality |
| Answer | B |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 142 |
| Question | Operator overloading is also called one form of polymorphism because\_\_\_\_\_\_\_. |
| A | the overloaded operators have many forms |
| B | the overloaded operators can be declared virtual |
| C | the overloaded function can perform various tasks depending upon the type of object |
| D | None of these |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 143 |
| Question | Overloading means |
| A | two or more methods in the same class that have same name |
| B | calling the method which has actual parameters |
| C | two or more methods having same name but present in different class |
| D | none of these |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 144 |
| Question | The inheritance mechanism provides meaning of deriving\_\_\_\_\_\_ |
| A | new operator from exciting one |
| B | new function from exciting one |
| C | new class from exciting one |
| D | all of these |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 145 |
| Question | A class derived from the exciting class is known as\_\_\_\_\_\_ |
| A | new class |
| B | Inheritee |
| C | derived class |
| D | none of these |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 146 |
| Question | The derived class is derived from\_\_\_\_\_\_\_\_\_\_ |
| A | derived class |
| B | base class |
| C | both a&b |
| D | none of these |
| Answer | B |
| Marks | 1 |
| Unit | 2 |

|  |  |
| --- | --- |
| **Id** | 147 |
| Question | Which of the following can be derived from base class in inheritace ? |
| A | data members |
| B | member function |
| C | both a&b |
| D | none of these |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 148 |
| Question | The inheritance is described as a \_\_\_\_\_ relationship |
| A | has a |
| B | is a |
| C | association |
| D | none of these |
| Answer | B |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 149 |
| Question | Which of the following allows you to create derived class that inherits properties from more than one base class ? |
| A | multilevel inheritance |
| B | multiple inheritance |
| C | single inheritance |
| D | Hybrid inheritance |
| Answer | B |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 150 |
| Question | The principle by which the knowledge of general category can be applied to more specific objects is called \_\_\_\_\_ |
| A | polymorphism |
| B | overriding |
| C | inheritance |
| D | none of these |
| Answer | A |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 151 |
| Question | Parent:child is\_\_\_\_\_\_\_\_ |
| A | base:derived |
| B | derived:driven |
| C | child:super |
| D | subclass:superclass |
| Answer | A |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 152 |
| Question | What is the syntax of inheritance of a class ? |
| A | Class class \_name |
| B | Class name:access specifier |
| C | Class name:access specifier class name |
| D | none of these |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 153 |
| Question | If an attribute is private then which methods have access to it? |
| A | Only static methods in the same class |
| B | Only the methods defined in that class |
| C | Only the methods of the same package |
| D | none of these |
| Answer | B |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 154 |
| Question | Which of the following advantage cannot be achieved by using multiple inheritance? |
| A | polymorphism |
| B | dynamic binding |
| C | both a&b |
| D | none of these |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 155 |
| Question | Which of the symbol used to create multiple inheritance ? |
| A | Dot |
| B | Comma |
| C | Hash # |
| D | Dollar |
| Answer | B |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 156 |
| Question | Using multiple inheritance |
| A | there can be virtual class |
| B | it can not be include virtual class |
| C | the base classes must have only default constructor |
| D | none of these |
| Answer | A |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 157 |
| Question | The\_\_\_\_\_\_member function is declared in base class but redefined in derived class |
| A | class |
| B | overloaded |
| C | operator |
| D | virtual |
| Answer | D |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 158 |
| Question | In public inheritance \_\_\_\_\_\_ |
| A | All the members of base class are inherited and are made public |
| B | Members of base class that are not private are inherited and retain their access type |
| C | All the members of base class are inherited and retain their access type |
| D | Only public members of base class are inherited and they remain public |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 159 |
| Question | If class C is derived class of class B and class B is a derived class of A. If we instantiate class B object then the first constructor called belongs to class |
| A | A |
| B | B |
| C | can be A or B |
| D | one cannot achieve such inheritance |
| Answer | A |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 160 |
| Question | When the object of derived class expire, first the \_\_\_\_\_\_\_ is invoked followed by the \_\_\_\_\_\_\_\_. |
| A | derived class constructor, base class destructor |
| B | derived class destructor , base class destructor |
| C | base class destructor , derived class destructor |
| D | none of these |
| Answer | B |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 161 |
| Question | If class A inherits from class B then B is called \_\_\_\_\_\_\_\_ ans A is called \_\_\_\_\_\_\_\_ of B. |
| A | superclass and subclass |
| B | subclass and superclass |
| C | subclass and child class |
| D | superclass and parent class |
| Answer | A |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 162 |
| Question | What does derived class does not inherit from the base class \_\_\_\_\_\_\_. |
| A | constructor and destructor |
| B | operator=() members |
| C | friends |
| D | all of these |
| Answer | D |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 163 |
| Question | Which constructor will initialise the base class data member ? |
| A | Base class |
| B | Derived class |
| C | Derived derived class |
| D | None of these |
| Answer | A |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 164 |
| Question | If class A is a friend class of class B, if class B is friend class of class C then\_\_\_\_\_\_\_ |
| A | class C is friend class of |
| B | class A is friend class of |
| C | class A and class C do not have any friendship relation. |
| D | none of these |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 165 |
| Question | \_\_\_\_\_\_\_\_\_ class is tightly coupled with other class. |
| A | friend |
| B | virtual |
| C | abstract |
| D | none of these |
| Answer | A |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 166 |
| Question | The keyword friend is used in \_\_\_\_\_\_\_\_\_\_\_. |
| A | the class allowing access to another class |
| B | the private section of a class |
| C | the public section of a class |
| D | all of these |
| Answer | D |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 167 |
| Question | Class Test:public A, public B is an example of multiple inheritance. |
| A | False |
| B | True |
| C |  |
| D |  |
| Answer | B |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 168 |
| Question | Which of the following interface determines how your program will be used by other program? |
| A | Public |
| B | Private |
| C | Protected |
| D | None of these |
| Answer | A |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 169 |
| Question | When base class pointer points to derived class object\_\_\_\_\_\_\_\_\_\_\_ |
| A | it can access only base class members |
| B | it can access only derived class members |
| C | both base class &derived class members |
| D | None of these |
| Answer | A |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 170 |
| Question | The base class will offer\_\_\_\_\_ |
| A | more specific object than the derived class |
| B | more generalized version of its derived class |
| C | empty templates of its derived class |
| D | none of these |
| Answer | B |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 171 |
| Question | In my program I have overloaded TEST::operator+ and TEST::operator= What is the effect on TEST::operator+= ? |
| A | The TEST::operator+= will be automatically overloaded .first TEST::operator+ will get overloaded and then TEST::operator= |
| B | The TEST::operator+= will be automatically overloaded .first TEST::operator= will get overloaded and then TEST::operator+ |
| C | TEST::operator+= will made invalid |
| D | There will be no effect because all three are independent |
| Answer | D |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 172 |
| Question | What will happen on execution of the following code ?  Class base  {  };class derived: protected base  {  }; |
| A | It will not compile as the class body of the base class is not defined |
| B | It will not compile as the class body of the derived class is not defined |
| C | It will compile successfully |
| D | The compilation of above code is dependent upon the type of data provided to it |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 173 |
| Question | The base class will offer\_\_\_\_\_ |
| A | more specific object than the derived class |
| B | more generalized version of its derived class |
| C | empty templates of its derived class |
| D | none of these |
| Answer | B |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 174 |
| Question | The hybrid inheritance is \_\_\_\_\_ |
| A | multiple inheritance |
| B | multilevel inheritance |
| C | multipath inheritance |
| D | both a &c |
| Answer | D |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 175 |
| Question | Ho  w many types of inheritance are there |
| A | 1 |
| B | 2 |
| C | 4 |
| D | 5 |
| Answer | D |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 176 |
| Question | Choose the correct option |
| A | a) a constructor can not be called explicitly |
| B | b) a destructor is not inherited |
| C | c) constructor can not be inherited |
| D | d) All of these |
| Answer | D |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 177 |
| Question | Suppose class derived is derived from a class Base. Both the classes contain the  Function name display() that take no argument. What will be the statement in the class derived which will called the display function of base class |
| A | Display() |
| B | Base:display() |
| C | Base ::display() |
| D | Can make such a cell |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 178 |
| Question | Suppose class derived is derived from a class Base privately. The object of class Derived is located in main() can access\_\_\_\_\_\_\_. |
| A | public members of base |
| B | private member of base |
| C | protected members of base |
| D | public members of derived |
| Answer | D |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 179 |
| Question | Multiple inheritance causes for a derived class to have \_\_\_members. |
| A | ambiguous |
| B | public |
| C | private |
| D | protected |
| Answer | A |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 180 |
| Question | What will be the first line of specifier for the class tier, wheel &rubber. Make use of public rubber |
| A | Class Tier:public wheel, public rubber |
| B | Class wheel:public tier, public rubber |
| C | Class rubber:public tier, public wheel |
| D | none of these |
| Answer | A |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 181 |
| Question | Which is the correct class defination for class C ,Which inherits from A &B classes |
| A | Class C:A,B |
| B | Class C::A,B |
| C | Class C:public A,public B |
| D | Class C:: public A,Public B |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 182 |
| Question | The ability of function or operator to act in different ways on different data type is called\_\_\_\_\_\_\_\_\_\_\_ |
| A | inheritance |
| B | polymorphism |
| C | encapsulation |
| D | none of these |
| Answer | B |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 183 |
| Question | \_\_\_\_\_class that declares or inherits a virtual function. |
| A | Encapsulation data |
| B | Inherited class |
| C | Polymorphic class |
| D | none of these |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 184 |
| Question | Choose the correct option. |
| A | A base class may have more then one derived class |
| B | Derived class may have more than one derived class |
| C | Both a &b |
| D | Neither a nor b |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 185 |
| Question | reusability is provided by which feature of c++ |
| A | polymorphisms |
| B | abstraction |
| C | derivation |
| D | none of the above |
| Answer | C |
| Marks | 1 |
| Unit | II |

|  |  |
| --- | --- |
| **Id** | 186 |
| Question | What types of derivations are supported by c++? |
| A | single |
| B | multiple |
| C | multilevel |
| D | all of the above |
| Answer | D |
| Marks | 1 |
| Unit | II |

Oop all mcqs

Unit I Fundamentals of Object Oriented Programming (06 Hours)

Introduction to procedural, modular, generic and object-oriented programming techniques,

limitations of procedural programming, Need of object-oriented programming, OOP Paradigms,

Fundamentals of object-oriented programming: Namespaces, objects, classes, data members,

methods, messages, data encapsulation, data abstraction and information hiding, inheritance,

polymorphism.

Benefits of OOP, C++ as object oriented programming language.

C++ Programming- C++ programming Basics, Data Types, Structures, Enumerations, control structures, Arrays and Strings, Class, Object, class and data abstraction, Access specifiers, separating

interface from implementation.

Functions- Function, function prototype, accessing function and utility function, Constructors and destructor, Types of constructor ,Objects and Memory requirements, Static members: variable and

functions, inline function, friend function.

Unit II Inheritance and Pointers (06 Hours) Inheritance- Base Class and derived Class, protected members, relationship between base Class and derived Class, Constructor and destructor in Derived Class, Overriding Member Functions, Class Hierarchies, Public and Private Inheritance, Types of Inheritance, Ambiguity in Multiple Inheritance, Virtual Base Class, Abstract class, Friend Class , Nested Class.

Pointers: declaring and initializing pointers, indirection Operators, Memory Management: new and delete, Pointers to Objects, this pointer, Pointers Vs Arrays, accessing Arrays using pointers, Arrays of Pointers, Function pointers, Pointers to Pointers, Pointers to Derived classes, Passing pointers to functions, Return pointers from functions, Null pointer, void pointer.

Unit III Polymorphism (06 Hours) Polymorphism- Introduction to Polymorphism, Early and late binding, Types of Polymorphism: Operator Overloading- concept of overloading, operator overloading, Overloading Unary Operators, Overloading Binary Operators, Data Conversion, Type casting (implicit and explicit), Pitfalls of Operator Overloading and Conversion, Keywords explicit and mutable. Function overloading Run Time Polymorphism- Pointers to Base class, virtual function and its significance in C++, pure virtual function and virtual table, virtual destructor, abstract base class.

Unit IV Files and Streams (06 Hours) Data hierarchy, Stream and files, Stream Classes, Stream Errors, Disk File I/O with Streams, File Pointers, and Error Handling in File I/O, File I/O with Member Functions, Overloading the Extraction and Insertion Operators, memory as a Stream Object, Command-Line Arguments, Printer output

Unit V Exception Handling & Templates (06 Hours) Exception Handling- Fundamentals, other error handling techniques, simple exception handlingDivide by Zero, Multiple catching, re-throwing an exception, exception specifications, user defined exceptions, processing unexpected exceptions, constructor, destructor and exception handling, exception and inheritance. Templates- , The Power of Templates, Function template, overloading Function templates, and class template, class template and Nontype parameters, template and friends Generic Functions, The typename and export keywords.

Unit VI Standard Template Library (STL) (06 Hours)

Introduction to STL, STL Components, Containers- Sequence container and associative containers,

container adapters, Application of Container classes: vector, list, Algorithms- basic searching and

sorting algorithms, min-max algorithm, set operations, heap sort, Iterators- input, output, forward,

bidirectional and random access. Object Oriented Programming – a road map to future

* Which of these is used to access members of class before the object of that class is created?  
  a) public  
  b) private  
  c) static  
  d) protected
* Which of these base classes are accessible to the derived class members?  
  a) static  
  b) protected  
  c) private  
  d) Shared
* What is the process by which we can control what parts of a program can access the members of a class?

a.Polymorphism

b.Abstraction

c.Encapsulation

d.Recursion

* What is an abstract class in C++?  
  a) Class specifically used as a base class with atleast one virtual functions  
  b) Class specifically used as a base class with atleast one pure virtual functions  
  c) Class from which any class is derived  
  d) Any Class in C++ is an abstract class

Explanation: An abstract class is defined as a class which is specifically used as a base class. An abstract class should have atleast one pure virtual function.

* Which is the correct syntax of defining a pure virtual function?  
  a) pure virtual return\_type func();  
  b) virtual return\_type func() pure;  
  c) virtual return\_type func() = 0;  
  d) virtual return\_type func();

Answer: c  
Explanation: virtual return\_type function\_name(parameters) = 0; where {=0} is called pure specifier.

* Which is the correct statement about pure virtual functions?  
  a) They should be defined inside a base class  
  b) Pure keyword should be used to declare a pure virtual function  
  c) Pure virtual function is implemented in derived classes  
  d) Pure virtual function cannot implemented in derived classes  
  Explanation: A pure virtual function does not have a definition corresponding to base class. All derived class may or may not have an implementation of a pure virtual function. there is no pure keyword in C++.
* Pick the correct statement.  
  a) Pure virtual functions and virtual functions are the same  
  b) Both Pure virtual function and virtual function have an implementation in the base class  
  c) Pure virtual function has no implementation in the base class whereas virtual function may have an implementation in the base class  
  d) The base class has no pure virtual function  
  Explanation: Pure virtual function has no implementation in the base class whereas virtual function may have an implementation in the base class. The base class has at least one pure virtual function
* 1. Which among the following is the main characteristic of class?

a) Inheritance

b) Encapsulation

c) Polymorphism

d) Object creation

Answer: b

Explanation: The classes are made to encapsulate the data and properties that are most similar and can be grouped together inside a single class. This single class can represent all of those features by creating its instances.

Which of the following is not recommended in a header file?

 **A.**  Type definitions (typedefs)

 **B**. Class definitions

 **C**. Function definitions

 **D.** Template definitions

Q. A function that does the same operation on different data types is to be implemented using \_\_\_\_\_\_\_\_\_\_.

A. macros

B. overloading

C. function templates

D. default arguments

Q. At what time a variable comes into existence in memory, is determined by its \_\_\_\_\_\_\_\_\_\_.

A. scope

B. storage class

C. data type

D. all of the above

Which of the following specifiers need not be honored by the compiler?

A.register

B.inline

C.static

D.Both (a) & (b

Option: D

Register and inline are not compiler directives but rather request to the compiler. These requests need not be honored by the compiler.

Q. Which of the following cannot be declared static?

A. Class

B. Object

C. Functions

D. Member variables

17: The order in which operands are evaluated in an expression is predictable if the operator is;

A.\*

B.+

C.%

D.&&

Option: D

Which of the following correctly describes C++ language?

A. Statically typed language

B. Dynamically typed language

C. Both statically and dynamically typed language

D. Type-less language

Q. Which of the following keyword supports dynamic method resolution?

A. abstract

B. virtual

C. dynamic

D. typeid

Correct Answer : OPTION B, virtual. The virtual keyword indicates that the virtual method will be resolved at runtime(i.e., the method resolution is dynamic

Q. Which of the following is the most common way of implementing C++?

A. C++ programs are directly compiled into native code by a compiler.

B. C++ programs are first compiled to intermediate code by a compiler and then executed by a virtual machine.

C. C++ programs are interpreted by an interpreter.

D. A C++ editor directly compiles and executes the programs.

Correct Answer : OPTION A, C++ programs are directly compiled into native code by a compiler.

What is the implicit pointer that is passed as the first argument for non-static member functions?

A. 'self' pointer

B. std::auto\_ptr pointer

C. 'myself' pointer

D. 'this' pointer

If X is the name of the class, what is the correct way to declare copy constructor of X?

A. X(X arg)

B. X(X\* arg)

C. X(const X\* arg)

D. X(const X& arg)

Correct Answer : OPTION D, X(const X& arg). The copy constructor takes a const reference to the class type as the argument.

Q. How many copies of a class's static member are shared between objects of the class?

A. A copy of the static member is shared by all objects of a class.

B. A copy is created only when at least one object is created from that class.

C. A copy of the static member is created for each instantiation of the class.

D. No memory is allocated for static members of a class.

Which of the following member functions is resolved dynamically?

A. static member function

B. const member function

C. virtual member function

D. non virtual member function

Which of the following type of class allows only one object of it to be created?

((OPTION\_A)) Virtual class

((OPTION\_B)) Abstract class

((OPTION\_C)) Singleton class

((OPTION\_D)) Friend class

Which of the following statements is correct?

((OPTION\_A)) Base class pointer cannot point to derived class

((OPTION\_B)) Derived class pointer cannot point to base class

((OPTION\_C)) Pointer to derived class cannot be created.

((OPTION\_D)) Pointer to base class cannot be create

Which of the following is not the member of class?

((OPTION\_A)) Static function

((OPTION\_B)) Friend function

((OPTION\_C)) Const function

((OPTION\_D)) Virtual function

Which of the following concepts means determining at runtime what

method to invoke?

((OPTION\_A)) Data hiding

((OPTION\_B)) Dynamic Typing

((OPTION\_C)) Dynamic binding

((OPTION\_D)) Dynamic loading

How many instances of an abstract class can be created?

((OPTION\_A)) 1

((OPTION\_B)) 5

((OPTION\_C)) 15

((OPTION\_D)) 0

Which of the following cannot be friend?

((OPTION\_A)) Function

((OPTION\_B)) Class

((OPTION\_C)) Object

((OPTION\_D)) Operator function

Which of the following concepts of OOPS means exposing only necessary

information to client?

((OPTION\_A)) Encapsulation

((OPTION\_B)) Abstraction

((OPTION\_C)) Data hiding

((OPTION\_D)) Data binding

What is the Standard Template Library?

a) Set of C++ template classes to provide common programming data structures and

functions

b) Set of C++ classes

c) Set of Template functions used for easy data structures implementation

d) Set of Template data structures only

2. Pick the correct statement.

a) STL is a generalized library

b) Components of STL are parameterized

c) STL uses the concept of templates classes and functions to achieve generalized

implementation

d) All of the mentioned

How many components STL has?

a) 1 b) 2

c) 3 d) 4

4. What are the containers?

a) Containers store objects and data

b) Containers stores all the algorithms

c) Containers contain overloaded functions

d) Containers contain set of Iterators

5. In how many categories, containers are divided?

a) 1

b) 2

c) 3

d) 4

6. What are the Sequence Containers?

a) Containers that implements data structures which can be accessed sequentially

b) Containers that implements sorted data structures for fast search in O(logn)

c) Containers that implements unsorted(hashed) data structures for quick search in O(1)

d) Containers that implements data structures which can be accessed non-sequentially

7. How many Sequence Containers are provided by C++?

a) 2

b) 3

c) 4

d) 5

What are the Associative Containers?

a) Containers that implements data structures which can be accessed sequentially

b) Containers that implements sorted data structures for fast search in O(logn)

c) Containers that implements unsorted(hashed) data structures for quick search in O(1)

d) Containers that implements data structures which can be accessed non-sequentially

What are Iterators?

a) Iterators are used to iterate over C-like arrays

b) Iterators are used to iterate over pointers

c) Iterators are used to point memory addresses of STL containers

d) Iterators are used to iterate over functions

15. Which header file is used for Iterators?

a) <iterator>

b) <algorithm>

c) <iter>

d) <loopIter>

16. What will be the output of the following C++ code?

#include <vector> #include <algorithm>

#include <iostream>using namespace std;

int main()

{vector<int> v;

for(int i=0;i<10;i++)

{v.push\_back(i+1);}

for(int i=0;i<10;i++)

{cout<<v[i]<<" ";

cout<<endl;

random\_shuffle(v.begin(), v.end());}

for(int i=0;i<10;i++)

{cout<<v[i]<<" ";}

return 0;

}

a)

1 2 3 4 5 6 7 8 9 10

5 4 8 9 1 6 3 2 7 10

17. What is the property of stable sort function provided by the STL algorithm?

a) sorts the elements of a sequence in ascending order preserving the relative order of equivalent

elements

b) sorts the elements of a sequence in descending order preserving the relative order of equivalent

elements

c) arranges the sequence randomly preserving the relative order of equivalent elements

d) same as sort function of STL algorithm

18. What is the property of partial sort function provided by the STL algorithm?

a) sorts the elements before the middle element in ascending order and remaining elements are left without any specific order

b) sorts the elements before the middle element in descending order and remaining elements are left without any specific order

c) sorts the elements after the middle element in ascending order and remaining elements are left without any specific order

d) sorts the elements after the middle element in descending order and remaining elements are left without any specific order

19. What will be the output of the following C++ code?

#include <iostream>

#include <algorithm>

#include <vector>

using namespace std;

int main ()

{

vector<int> v = {4,2,10,5,1,8};

sort(v.begin(), v.end());

if (binary\_search(v.begin(), v.end(), 4))

cout << "found.\n";

else

cout << "not found.\n";

return 0;

}

a) found.

b) not found.

c) Error

d) Segmentation fault

20. What will be the output of the following C++ code?

#include <iostream>

#include <algorithm>

#include <vector>

using namespace std;

int main ()

{

vector<int> v = {4,2,10,5,1,8};

if (binary\_search(v.begin(), v.end(), 4))

cout << "found.\n";

else

cout << "not found.\n";

return 0;

}

a) found.

b) not found.

c) Error

d) Segmentation fault

21. Which function can be used to find the sum of a vector container?

a) findsum()

b) accumulate()

c) calcsum()

d) checksum()

22. Which header file is required to use accumulate() function?

a) <algorithm>

b) <numeric>

c) <vector>

d) <iostream>

What is meaning of template parameter?

a)It is used to pass a type as argument

b)Used to evaluate a type

c)It can of no return type

d)None of the mentioned

\_\_\_\_\_\_Keyword is used in template.

Class

Typename

Both a and b

Using

What is scope of template parameter?

Inside a block only

Inside the class only

Throughout program

All of the above

Function overloading is also similar to which of the following

Operator overloading

Destructor overloading

Constructor overloading

Virtual function

Template are of types

Function template

Class template

Both a and b

None of the above

Class template can be created using\_\_\_\_\_\_\_\_syntax.

Template<class T>class classname

Template<class T1,class T2> class classname

Both a and b

None of the above mentioned

Syntax for creating a function template is

A.Template<typename t>returntype function name

B.Template<class T> returntype function name

C.Both a and b

D.None of the above mentioned

Pick up the correct statement

i)template allow us to define generic classes and functions

ii)template support generic programming

iii)function template overloading is possible

A.i only

B.i and ii only

C.ii and iii only

D.i, ii and iii

Template function can be overloaded

A.True

B.False Why we use :: template-template parameter?

A.binding

B.rebinding

C.both a &b

D.none of these

Which of the things does not require instantiation?

A.functions

B.non virtual member function

C.member class

D.all of the mentioned

A template provides a convenient way to make a family of

A.variables.

B.functions

C.classes

D.B and C

Templates automatically create different versions of a function, depending on user input.

A.TRUE

B.FALSE A template class

A.is designed to be stored in different containers.

B.works with different data types.

C.generates objects which must all be identical.

D.generates classes with different numbers of member functions.

There can be more than one template argument.

A.TRUE

B.FALSE

Actual code for a template function is generated when

A.the function declaration appears in the source code.

B.the function definition appears in the source code.

C.a call to the function appears in the source code.

D.the function is executed at runtime.

An exception is typically caused by

A.the programmer who writes an application’s code.

B.the creator of a class who writes the class member functions.

C.a runtime error.

D.an operating system malfunction that terminates the program.

Statements that might cause an exception must be part of a catch block.

A.TRUE

B.FALSE

Exceptions are thrown

A.from the catch block to the try block.

B.from a throw statement to the try block.

C.from the point of the error to a catch block.

D.from a throw statement to a catch block.

A statement that throws an exception does not need to be located in a try block.

A.TRUE

B.FALSE The following is/are errors for which an exception would typically be thrown:

A.An excessive amount of data threatens to overflow an array.

B.new cannot obtain the requested memory.

C.A power failure shuts down the system.

D.A and B

Additional information sent when an exception is thrown may be placed in

A.the throw keyword.

B.the function that caused the error.

C.the catch block.

D.an object of the exception class. A program can continue to operate after an exception has occurred.

A.TRUE

B.FALSE

What is the output of following program?

#include <iostream> using namespace std;

int main()

{int x = -1;

try {

cout <<"Inside try \n";

if (x <0){

throw x;

cout <<"After throw \n";

}}

catch (int x ) {

cout <<"Exception Caught \n";

}

cout <<"After catch \n";

return 0;

}

B.Inside try

Exception Caught

After catch

What is the advantage of exception handling?

1) Remove error-handling code from the software's main line of code.

2) A method writer can chose to handle certain exceptions and delegate others to the caller.

3) An exception that occurs in a function can be handled anywhere in the function call stack.

A.Only 1

B.1, 2 and 3

C.1 and 3

D.1 and 2

What should be put in a try block?

1. Statements that might cause exceptions

2. Statements that should be skipped in case of an exception

A.Only 1

B.Only 2

C.Both 1 and 2

D.None of the above

What is the output of following program

#include<iostream> using namespace std;

class Base {};

class Derived: public Base {};

int main()

{Derived d;

try {

throw d; }

catch(Base b) {

cout<<"Caught Base Exception";}

catch(Derived d) {

cout<<"Caught Derived Exception";}

return 0;

}

A.Caught Derived Exception

B.Caught Base Exception

C.Caught Derived Exception and Caught Base Exception

D.Compiler Error

What is the output of following program?

#include <iostream> using namespace std;

int main()

{ try

{throw 10; }

catch (...)

{ cout <<"default exception\n"; }

catch (int param)

{cout <<"int exception\n";}

return 0;

}

A.default exception

B.int exception

C.default exception int exception

D.Compiler Error

What is the output of following program?

#include <iostream> using namespace std;

class Test {

public:

Test() { cout <<"Constructing an object of Test "<<endl; }

~Test() { cout <<"Destructing an object of Test " <<endl; }

};

int main() {

try {Test t1;

throw 10;

} catch(int i) {

cout <<"Caught "<<i <<endl;

}

}

A.Caught 10

B.Constructing an object of Test

Caught 10

C.Constructing an object of Test

Destructing an object of Test

Caught 10

D.Compiler Errror

What happens in C++ when an exception is thrown and not caught anywhere like following program?

#include <iostream>

using namespace std;

int fun() throw (int)

{ throw 10;

}

int main() {

fun();

return 0;

}

A.Compiler error

B.Abnormal program termination

C.Program doesn't print anything and terminates normally

D.None of the above

Which alternative can replace the throw statement ?

A.Exit

B.For

C.Break

D.Return

Which of the following keyword can not be appered inside the class?

A.Virtual

B.Static

C.Template

D.Friend

What is template?

A.Template is formula for creating a generic class

B.Template is used to manipulate class

C.Template is used for creating functions

D.None of these

Select the correct syntax of template:

A.Template

B.Template<>

C.Temp

D.None of these

A class is generated from template class is called \_\_\_\_\_\_\_.

A.inherited class

B.derived class

C.generated class

D.subclass

\_\_\_\_\_\_\_\_ is useful when template of template is used?

A.Friend function

B.Static function

C.Typedef

D.Inheritance

Which of the C++ feature allows you to create classes that are dynamic for using data types?

A.Templates

B.Inheritance

C.Polymorphism

D.Information hiding

A function template means \_\_\_\_\_\_\_.

A.creating a function having exact type

B.creating a function without having to specify exact type

C.both a and b

D.none of these

Which of the following is used to describe the function using placeholder type?

A.Template type parameter

B.Template parameter

C.Template type

D.None of these

String template is used \_\_\_\_\_.

A.to replace a string.

B.to replace a string with another string

C.to delete a string

D.none of these

Maximum number of template argument in function template is \_\_\_\_\_\_\_.

A.two

B.three

C.four

D.many

Template function must have

A.one or more than one argument

B.zero argument

C.only one argument

D.at least two arguments

Template function must have at least \_\_\_\_\_\_\_\_ generic data type.

A.zero

B.one

C.two

D.none of these

Templates provide way of abstracting \_\_\_\_\_\_ information.

A.type

B.data

C.method

D.access

If you create instantiation of a class template with an int and then create a second instantiation with a double then

A.once the function is used for one data type it becomes unavailable for other type

B.you can not perform this kind of operation in C++

C.you must precede each function call with the word int or double

D.none of these

If templates were removed from C++,Which of the following will be true?

I. Some algorithms could no longer be implemented

II. Any particular algorithms could still be implemented but often less elegantly.

A.Only I is true

B.Only II is true

C.Both I and II is true

D.None of these

In the template <class T>declaration of T stands for \_\_\_\_\_\_\_\_.

A.integer data type

B.arbitary class

C.generic data types

D.none of these

What is the meaning of template parameter?

A.It is used to pass a type as argument

B.It is used to evalute a type

C.It has no return type

D.None of these

What can be passed by non-type template parameter during compile time?

A.Int

B.Double

C.Char

D.constant expression

Choose the correct statement from the following:

A.Template function will take long time to execute

B.Template functions are written when you want to have only one code for many different types

C.due to template function the duplicate code will get increased

D.None of these

How many types of templates are there in c++?

A.Two

B.Three

C.Four

D.None Of These

What is the task of compiler while handling template?

A.type association

B.Portability

C.code elimination

D.all of the above

What should be the name of the parameter that the template should take?

A.same as class

B.same as function

C.same as template

D.none of thesE Which keyword can be used with template?

ATypename

B.Operator

C..both a and b

D.None of these

Which of the following describes a difference between template function and

template class in c++?

A.The compiler determines the type of a template function's arguments, but the types of template classes must be stated explicitly when declaring objects

B.template functions cannot be defined for user-defined types, but template classes can

C.template classes cannot be defined for user-defined types,but templatefunctions can.

D.None Of These

What is the validity of templet parameter?

A.Inside the class

B.Inside the block

C.whole program

D.None of these

Which of the following does not required installation ?

A.Non virtual member function

B.Member class

C..Function

D.All of above

Which keyword is used to handle the exception ?

A.Try

B.Catch

C.Throw

D.Exception

What is the use of the keyword finally ?

A.It is used at the start of the program for handling all the exceptions

B.It is used at the end of the program to handle all the exceptions

C.It can be used anywhere in the program to handle all the exceptions

D.None of these Which of the following most preferred way of throwing and handling exception?

A.Throw by value and catch by reference

B.Throw by reference and catch by value

C.Throw by value and catch by value

D.None of these

Which of the following is the most general exception handler that catches exception of any type?

A.Catch(std::exception)

B.Catch(std:any\_exception)

C.Catch(...)

D.Catch()

Which of the following causes an exception

A.Missing parenthesis in main()

B.Calling a function which is not present

C.A syntax error

D.a run time error

Which block should be placed after try block ?

A.Throw

B.Catch

C.both a or b

D.none of these

Choose the correct statement

A.Exception are not suitable for critical points in the program

B.Exception are suitable for critical points in the program

C.Both a&b

D.None of these

In C++ program handling, a try block must be followed by \_\_\_\_\_catch blocks

A.exactly one

B.one or more

C.exactly two

D.none of these

The process of handling the actual exception occurs \_\_\_\_\_\_\_\_\_

A.inside the program

B.outside the program

C.both a &b

D.none of these

Which of the following is used to check the error in the block?

A.Try

B.Throw

C.Catch

D.None of these

What should be present when throwing object ?

A.Constructor

B.Destructor

C.copy constructor

D.none of these

For handling the exception in C++ \_\_\_\_\_\_\_ are used

A.catch handlers

B.exception handlers

C.Pointers

D.none of these

For handling the exceptions in C++ \_\_\_\_\_\_\_\_\_ is used .

A.handler function

B.terminate function

C.both a &b

D.none of these

How many parameters does the throw expression can have ?

A.0

B.1

C.2

D.3

What kind of exceptions are used in C++

A.Handled

B.Unhandled

C.Static

D.Dynamic

What will happen when exception is uncaught?

A.Arise an error

B.program will run

C.execute in a loop

D.none of these

Choose the correct statement

A.A function can throw any type of exception

B.a function can throw an exception of certain type only

C.A exception can't throw any type of exception

D.none of thesE

What function will be called when we have uncaught exception?

A.Catch

B.Throw

C.Terminate

D.none of these

What will happen when a programs throws any other of exception other than specified ?

A.still execute

B.Terminate

C.raise an error

D.none of these.

Which statement is used to catch all types of exceptions?

A.catch()

B.catch(Test t)

C.catch

D.None of these

Which keyword can be used as a template

A.Exception

B.Typename

C.both a & b

D.Function

An Exception is thrown using \_\_\_\_\_\_\_\_\_\_\_\_\_keyword in cpp

A.Throws

B.Throw

C.Threw

D.Thrown

Which parameter is legal for non-type template?

A.pointer to member

B.object

C.class

D.none of these Which of the things does not require instantiation?

A.functions

B.Non virtual member function

C.member class

D.all of these

Which of the following permits function overloading on c++?

A.Data Type

B.Number of arguments

C.A &B both

D.none of these

Function overloading is also similar to which of the following?

A.Operator Overloading

B.Constructer overloading

C.Destructor overloading

D.none of the above

Which is dependent on template parameter

A.base class

B.abstract class

C.method

D.none of the above

Which is used to handle the exceptions in c++?

A.catch handler

B.handler

C.exception handler

D.all of these

Which is called on allocating the memory for array of objects?

A.Function

B.Method

C.Destructor

D.Constructor

Which value is placed in the base class?

A.Inherited value

B.Derived value

C.Default type values

D.Both a and b

\_\_\_\_\_\_\_\_\_\_is used to perform the generic programming.

A.Class

B.Template

C.Function

D.Inheritance

A template can be considered as a kind of macros

A.True

B.False

We can not define more than 2 placeholder in class/function template.

A.False

B.True

When template is defined with parameter that would be replaced by specified \_\_\_\_\_\_\_at the time of actual use of class or function.

A.Keyword

B.Operator

C.Datatype

D.None of the above mentioned Templates sometimes called as \_\_\_\_\_\_\_\_\_\_\_

A.Parameterized classes

B.Parameterized function

C.Both a and b

D.None of the above mentioned

Exceptions are of type

A.Synchronous

B.Asynchronous

C.Both a and b

D.None of the above mentioned

“out-of-range”, “overflow” are the type of exceptions

A.Asynchronous

B.Synchronous

C.Default

D.None of the above

The most type of error--------.

A.Logical error

B.Syntactic error

C.Both a and b

D.Class

Run time error is known as \_\_\_\_\_\_

A.Logical error

B.Syntactic error

C.Exception

D.All of the above mentioned

How the exception is throw

A.throw exception

B.throw(exception)

C.throw

D.All of the above

Can we throw exception more than one time

A.True

B.False

Which statement we have to use rethrowing exception

A.throw(exception)

B.Throw

C.Both a and b

D.None of the above mentioned

Question Exception can be handle if\_\_\_\_\_\_\_

A.Throwing argument is match with catch block

B.Throwing argument is not match with catch block

C.Exception is not thrown

D.None of the abve

With this concept same algorithm can be used for different data types

A.Procedure oriented paradigm

B.Generic programming

C.Both a and b

D.None of the abovE

Template is a way creating generalize functions and classes which are applicable for all data types

A.False

B.True

Class template is applicable for \_\_\_.

A.For function only

B.For that class only

C.Both a and b

D.None of the above mentioned

Function template is applicable for\_\_\_\_\_\_\_\_\_.

A.For function only

B.For that class only

C.Both a and b

D.None of the above mentioned

How many kinds of parameters are there in c++

A.1

B.2

C.3

D.4

Which type of program is recommended to include in try block

A.Static memory allocation

B.Dynamic memory allocation

C.Const reference

D.Pointer

Answer How to handle error in destructor

A.Throwing

B.Terminate

C.Both a and b

D.None of the mentioned

In catch statement we have multiple parameters

A.Yes

B.No

------kind of exceptions are in c++.

A.Handled

B.Static

C.Both a and b

D.Unhandled

Pick up the correct statement

A.To throw exception we have to use catch statement

B.Error occurring code is placed in try block

C.We can not have multiple throwing mechanism in c++

D.Both and b

Answer

Can we used constructor for exception handling

A.Yes

B.No

Class template can be overloaded

A.True

B.False

\_\_\_\_\_\_is a generic class handler

A.Catch(---)

.B.Catch(-,-)

C.Catch(…)

D.Catch(void)

If the exception is not handled then which standard library function get invoked

A.stop()

B.terminate()

C.Read()

D.Write()

Exception can be only built in type

A.True

B.False

What will be output of program

#include<iostream>

using namespace std;

template<class T>

T display(T x)

{cout<< “using template x=”<<x<<“\n”;}

int display(int x)

{cout<<“Normal display x=”<<x <<“\n”;}

int main()

{display(2.3);

display(3);

diplay(1.1);}

B.using template x=2.3

Normal display x=3

using template x=1.1

In nested try blocks, there is no need to specify catch handler for inner try block. Outer catch handler is sufficient for the program

ATrue

BFalse

Can we write try block within try block

ATrue

BFalse

Can we prevent a function from throwing any exceptions

AYes

BNo

What is return type of uncaught\_exception() is----

A.Char\*

B.Double

C.Int

D.Bool

Can we write a throw statement inside catch statement

A.Yes

B.No

We can define our own exceptions in c++

AFalse

BTrue

Stack unwinding deals with

A.Polymorphism

B.inheritance

C.Exception handling

D.Classes

Pick up the correct statement

•Catch statement be placed immediately after try block

•It can have multiple parameters

•There must be multiple catch handler for a try block

•Generic catch statement we can placed anywhere in program

A.i and ii

B.i and iii

C.i and iv

D.i , ii and iii

Generic catch should be placed at

A.End of all statement

B.Before try

C.Before throw

D.Inside try

Irrespective of exception occurrence, catch handler will be always executed

A.Yes

B.No

From where does the template class can derived

A.Regular non-templated c++ class

B.Templated class

C.Both a and b

D.None of the above mentioned

What is done by compiler for templates

A.Type-safe

B.Code elimination

C.Portability

D.All of the above mentioned Catch handler itself may detect and throw an exception

A.True

B.False

If the thrown exception will not be caught by any catch statement then it will be passed to next outer try/catch sequence for processing.

AFalse

BTrue

While specifying the exceptions, the type-list specifies the\_\_\_\_\_\_\_\_ that may be thrown.

A.How many exceptions

B.Type of exception

C.Both a and b

D.None of the above mentioned

When an exception is rethrown ,it will not be caught by the\_\_\_\_\_\_\_\_\_\_or other catch in that group.

A.Same catch

B.Nested catch

C.Both a and b

D.None of the above mentioned

Try block can throw any exception

A.True

B.False

Pick up the correct statement from the following

1. Multiple catch statement are there in c++.
2. We have generic catch statement to handle all type of exception
3. Try block is used to throw and exception

A.i and iii

B.i

C.ii

D.i and ii only

When an exception is not caught

A.Program is go in wait condition

B.Program is aborted

C.Program works fine way

D.None of the above mentioned

We can place two or more catch blocks together to catch and handle multiple types of exceptions thrown by a try blocks

A.True

B.False

It is also possible to make a catch statement to catch all types of exceptions using ellipses as its arguments

A.True

B.False

We can restrict a function to throw only a set of specified exceptions by adding a throw specification clause to the function definition.

A.True

B.False

We may also use non-type parameters such basic or derived data types as arguments template

ATrue

B.False

Pick up the correct statement from the following related with overloading of template functions

•Call an ordinary function that has an exact match

•Call a template function that could be created with an exact macth

•Try normal overloading resolution to ordinary functions and call the one that matches

A.1 and 2 only

B.2 and 3 only

C.All of the above

D.None of the above mentioned What will be output of the a following program

#include<iostream>

using namespace std;

template <class T>

void display(T x)

{cout<<“Template display:”<<x<< “\n”;}

void display(int x)

{cout<<“Explicit display:”<<x <<“\n”;}

int main()

{display(100);

display(12.34);

display(‘c’);}

B.Explicit display:100

Template display:12.34

Template display: c

What will be output of program

#include <iostream>

using namespace std;

int main()

{cout <<"Start\n";

try {

cout <<"Inside try block\n";

throw 100;

cout << "This will not execute";

}

catch (int i) {

cout <<"Caught an exception -- value is: ";

cout <<i <<"\n";

}

cout <<"End";

return 0;

}

A.Start

Inside try block

Caught an exception -- value is: 100

End

What will be output of following program

#include <iostream>

using namespace std;

void Xhandler(int test)

{

try{

if(test) throw test;

else throw "Value is zero";

}

catch(int i) {

cout << "Caught Exception #: " <<i <<'\n';

}

catch(const char \*str) {

cout <<"Caught a string: ";

cout << str <<'\n';

}

}

int main()

{

cout <<"Start\n";

Xhandler(1);

Xhandler(2);

Xhandler(0);

Xhandler(3);

cout << "End";

return 0;

}

C.Start

Caught Exception #: 1

Caught Exception #: 2

Caught a string: Value is zero

Caught Exception #: 3

End

What will be output of program

#include <iostream>

using namespace std;

void Xhandler(int test)

{

try{

if(test==0) throw test; // throw int

if(test==1) throw 'a'; // throw char

if(test==2) throw 123.23; // throw double

}

catch(int i) { // catch an int exception

cout <<"Caught an integer\n";

}

catch(...) { // catch all other exceptions

cout <<"Caught One!\n";

}

}

int main()

{

cout <<"Start\n";

Xhandler(0);

Xhandler(1);

Xhandler(2);

cout <<"End";

return 0;}

C.Start

Caught an integer

Caught One!

Caught One!

End

What will be output of following program

#include <iostream>

using namespace std;

template <class Type1, class Type2> class myclass

{

Type1 i;

Type2 j;

public:

myclass(Type1 a, Type2 b) { i = a; j = b; }

void show() { cout <<i << ' ' <<j <<'\n'; }

};

int main()

{

myclass<int, double>ob1(10, 0.23);

myclass<char, char \*>ob2('X', "Templates add power.");

ob1.show(); // show int, double

ob2.show(); // show char, char \*

return 0;

}

A.10 0.23

X Templates add power.

B.0.23 10

X Template add power

C.10 10

X template add power

D.Compilation error

We can combine operator overloading with a class

A.True

B.False

If you overload a generic function, that overloaded function overrides (or "hides") the generic function relative to that specific version.

A.True

B.False

What will be output of following programming

#include <iostream>

using namespace std;

template <class T>

T GetMax (T a, T b) {

T result;

result = (a>b)? a : b;

return (result);

}

int main () {

int i=5, j=6, k;

long l=10, m=5, n;

k=GetMax<int>(i,j);

n=GetMax<long>(l,m);

cout <<k << endl;

cout <<n <<endl;

return 0;

}

A.6

10

B.5

5

C.10

10

D.Compilation error

What will be output of following program

#include <iostream>

using namespace std;

template <class T>

class mypair {

T a, b;

public:

mypair (T first, T second)

{a=first; b=second;}

T getmax ();

};

template <class T>

T mypair<T>::getmax ()

{

T retval;

retval = a>b? a : b;

return retval;

}

int main () {

mypair <int>myobject (100, 75);

cout << myobject.getmax();

return 0;

}

A.75

B.100

C.75

100

D.Compilation error

What will be output of following program

#include <iostream>

#include <exception>

using namespace std;

class myexception: public exception

{

virtual const char\* what() const throw()

{

return "My exception happened";

}

} myex;

int main () {

try

{

throw myex;

}

catch (exception&e)

{

cout << e.what() <<endl;

}

return 0;

}

A.Exception happened

B.My exception happened.

C.Run Time error

D.Compilation error

Pick up the correct statement from following

1.Exception handling is not supported c++

2.Template support generic programming in c++

3.overloading of function template is possible in c++

4.generic catch template can handle all types of exceptions

A.2 and 3 only

B.3 and 4 only

C.1, 2 and 3 only

D.2, 3 and 4 only

What does the following statement mean?

int (\*fp)(char\*)

A.pointer to a pointer

B.pointer to an array of chars

C.pointer to function taking a char\* argument and returns an int

D.function taking a char\* argument and returning a pointer to int

The operator used for dereferencing or indirection is \_\_\_\_

\*

&

->

–>>

Choose the right option

string\* x, y;

A.x is a pointer to a string, y is a string

B.y is a pointer to a string, x is a string

C.both x and y are pointer to string types

D.none of the mentioned

Which one of the following is not a possible state for a pointer?

A.hold the address of the specific object

B.point one past the end of an object

C.Zero

D.point to a byte

Which of the following is illegal?

A.int \*ip;

B.string s, \*sp = 0;

C.int i; double\* dp = &i;

D.int \*pi = 0;

#include <iostream>

using namespace std;

int main()

{

int a = 5, b = 10, c = 15;

int \*arr[ ] = {&a, &b, &c};

cout <<arr[1];

return 0;

}

A.10

B.15

C.20

D.Random number

The correct statement for a function that takes pointer to a float, a pointer to a pointer to a char and returns a pointer to a pointer to a integer is

A.int \*\*fun(float\*\*, char\*\*)

B.int \*fun(float\*, char\*)

C.int \*\*\*fun(float\*, char\*\*)

D.int \*\*\*fun(\*float, \*\*char)

#include <iostream>

using namespace std;

int main()

{

char arr[20];

int i;

for(i = 0; i < 10; i++)

\*(arr + i) = 65 + i;

\*(arr + i) = '\0';

cout << arr;

return(0);

}

A.ABCDEFGHIJ

B.AAAAAAAAA

C.JJJJJJJJJJJJ

D.None

#include <iostream>

using namespace std;

int main()

{

char \*ptr;

char Str[] = "abcdefg";

ptr = Str;

ptr += 5;

cout << ptr;

return 0;

}

A.fg

B.cdef

C.defg

D.abcd

Which rule will not affect the friend function?

A.private and protected members of a class cannot be accessed from outside

B.private and protected member can be accessed anywhere

C.both a &b

D.None

#include <iostream>

using namespace std;

class Box

{

double width;

public:

friend void printWidth( Box box );

void setWidth( double wid );

};

void Box::setWidth( double wid )

{

width = wid;

}

void printWidth( Box box )

{

box.width = box.width \* 2;

cout << "Width of box : " << box.width << endl;

}

int main( )

{

Box box;

box.setWidth(10.0);

printWidth( box );

return 0;

}

A.40

B.5

C.10

D.20

Pick out the correct statement.

A.A friend function may be a member of another class.

B.A friend function may not be a member of another class.

C.A friend function may or may not be a member of another class.

D.None of the mentioned

Where does keyword ‘friend’ should be placed?

A.function declaration

B.function definition

C.main function

D.None

#include <iostream>

using namespace std;

class sample

{

private:

int a, b;

public:

void test()

{

a = 100;

b = 200;

}

friend int compute(sample e1);

};

int compute(sample e1)

{

return int(e1.a + e1.b) - 5;

}

int main()

{

sample e;

e.test();

cout << compute(e);

return 0;

}

A.100

B.200

C.300

D.295

#include <iostream>

using namespace std;

class base

{

int val1, val2;

public:

int get(){

val1 = 100;

val2 = 300;}

friend float mean(base ob);

};

float mean(base ob)

{ return float(ob.val1 + ob.val2) / 2; }

int main()

{base obj;

obj.get();

cout << mean(obj);

return 0;

}

A.200

B.150

C.100

D.300

To which does the function pointer point to?

A.Variable

B.Constants

C.Function

D.absolute variables

What we will not do with function pointers?

A.allocation of memory

B.de-allocation of memory

C.both a &b

D.None

#include <iostream>

using namespace std;

int add(int first, int second)

{return first + second + 15;}

int operation(int first, int second, int (\*functocall)(int, int))

{return (\*functocall)(first, second); }

int main()

{

int a;

int (\*plus)(int, int) = add;

a = operation(15, 10, plus);

cout << a;

return 0;

}

A.25

B.36

C.40

D.45

#include <iostream>

using namespace std;

void func(int x)

{cout << x ; }

int main()

{void (\*n)(int);

n = &func;

(\*n)( 2 );

n( 2 );

return 0;

}

A.2

B.21

C.22

D.20

#include <iostream>

using namespace std;

int n(char, int);

int (\*p) (char, int) = n;

int main()

{ (\*p)('d', 9);

p(10, 9);

return 0;

}

int n(char c, int i)

{cout << c << i;

return 0;

}

A.d99

B. d9d9

C.d9

D.Compile time error

#include <iostream>

using namespace std;

int func (int a, int b)

{cout << a;

cout << b;

return 0;

}

int main(void)

{

int(\*ptr)(char, int);

ptr = func;

func(2, 3);

ptr(2, 3);

return 0;

}

A.2323

B.232

C.23

D.Compile time error

What are the mandatory part to present in function pointers?

A&

B.return values

C.Data types

D.None

What is meaning of following declaration?

int(\*ptr[5])();

A.ptr is pointer to function.

B.ptr is array of pointer to function.

C.ptr is pointer to such function which return type is array.

D.ptr is pointer to array of function.

What is size of generic pointer in c?

A.0

B.1

C.2

D.Null

Void pointer can point to which type of objects?

A.Int

B.Float

C.Double

D.All

What does the following statement mean?

int (\*fp)(char\*)

A.pointer to a pointer

B.pointer to an array of chars

C.pointer to function taking a char\* argument and returns an int

D.function taking a char\* argument and returning a pointer to int

#include <iostream>

using namespace std;

int main()

{int a[2][4] = {3, 6, 9, 12, 15, 18, 21, 24};

cout << \*(a[1] + 2) << \*(\*(a + 1) + 2) << 2[1[a]];

return 0;

}

A.15 18 21

B.21 21 21

C.24 24 24

D.Compile time error

#include <iostream>

using namespace std;

int main()

{ int i;

char \*arr[] = {"C", "C++", "Java", "VBA"};

char \*(\*ptr)[4] = &arr;

cout << ++(\*ptr)[2];

return 0;

}

A.ava

B.java

C.c++

D.Compile time error

#include <iostream>

using namespace std;

int main()

{int arr[] = {4, 5, 6, 7};

int \*p = (arr + 1);

cout << \*p;

return 0;

}

A.4

B.5

C.6

D.7

#include <iostream>

using namespace std;

int main()

{

int arr[] = {4, 5, 6, 7};

int \*p = (arr + 1);

cout << arr;

return 0;

}

A.4

B.5

C.Address of arr

D.7

#include <iostream>

using namespace std;

int main ()

{int numbers[5];

int \* p;

p = numbers; \*p = 10;

p++; \*p = 20;

p = &numbers[2]; \*p = 30;

p = numbers + 3; \*p = 40;

p = numbers; \*(p + 4) = 50;

for (int n = 0; n < 5; n++)

cout << numbers[n] << ",";

return 0;

}

A.10,20,30,40,50,

B.1020304050

C.Compile time error

D.Runtime error

#include <iostream>

using namespace std;

int main()

{

int arr[] = {4, 5, 6, 7};

int \*p = (arr + 1);

cout << \*arr + 9;

return 0;

}

A.12

B.5

C.13

D.Error

A void pointer cannot point to which of these?

A.methods in c++

B.class member in c++

C.all of the mentioned

D.None

#include <iostream>

using namespace std;

int func(void \*Ptr);

int main()

{char \*Str = "abcdefghij";

func(Str);

return 0;

}

int func(void \*Ptr)

{cout << Ptr;

return 0;}

A.abcdefghij

B.address of string “abcdefghij”

C.Compile time

D.Run time error

#include <iostream>

using namespace std;

int main()

{int \*p;

void \*vp;

if (vp == p);

cout << "equal";

return 0;

}

A.Equal

B.No output

C.Compile time error

D.Run time error

#include <iostream>

using namespace std;

int main()

{int n = 5;

void \*p = &n;

int \*pi = static\_cast<int\*>(p);

cout << \*pi << endl;

return 0;

}

A.5

B.6

C.Compile time error

D.Run time error

#include <iostream>

using namespace std;

int main()

{int a = 5, c;

void \*p = &a;

double b = 3.14;

p = &b;

c = a + b;

cout << c << '\n' << p;

return 0;

}

A.8, memory address

B.8.14

C.memory address

D.None

What we can’t do on a void pointer?

A.pointer arithemetic

B.pointer functions

C.Both

D.None

Which value we cannot assign to reference?

A.Integer

B.Floating

C.Unsigned

D.Null

#include <iostream>

using namespace std;

int main()

{int a = 9;

int & aref = a;

a++;

cout << "The value of a is " << aref;

return 0;

}

A.9

B.10

C.11

D.Error

#include <iostream>

using namespace std;

void print (char \* a)

{

cout << a << endl;

}

int main ()

{const char \* a = "Hello world";

print(const\_cast<char \*> (a) );

return 0;

}

A.Hello world

B.Hello

C.World

D.Compile time error

Identify the correct sentence regarding inequality between reference and pointer.

A.we can not create the array of reference.

B.we can create the Array of reference.

C.we can use reference to reference.

D.None

Which is used to tell the computer that where a pointer is pointing to?

A.Dereference

B.Reference

C.heap operations

D.None

#include <iostream>

using namespace std;

int main()

{

int x;

int \*p;

x = 5;

p = &x;

cout << \*p;

return 0;

}

A5

B.10

C.Memory address

D.None

#include <iostream>

using namespace std;

int main()

{

int x = 9;

int\* p = &x;

cout << sizeof(p);

return 0;

}

A.4

B.2

C.Depends on compiler

D.None

#include <iostream>

using namespace std;

int main()

{

double arr[] = {5.0, 6.0, 7.0, 8.0};

double \*p = (arr+2);

cout << \*p << endl;

cout << arr << endl;

cout << \*(arr+3) << endl;

cout << \*(arr) << endl;

cout << \*arr+9 << endl;

return 0;

}

A.7

0xbf99fc98

8

5

14

B.7

8

0xbf99fc98

5

14

C.0xbf99fc98

D.None

What does the dereference operator will return?

A.rvalue equivalent to the value at the pointer address.

B.lvalue equivalent to the value at the pointer address.

C.it will return nothing

D.None

Which operator is used in pointer to member function?

A.\*

B.->\*

C.Both a &b

D.None

#include <iostream>

using namespace std;

class Foo

{public:

Foo(int i = 0){ \_i = i;}

void f()

{cout << "Executed"<<endl;}

private:

int \_i;

};

int main()

{Foo \*p = 0;

p -> f();

}

A.Executed

B.Error

C.Run time error

D.None

Which is the best design choice for using pointer to member function?

A.Interface

B.Class

C.Structure

D.None

Virtual functions allow you to

A.create an array of type pointer-to-base class that can hold pointers to derived classes.

B.create functions that can never be accessed.

C.group objects of different classes so they can all be accessed by the same function code.

D.use the same function call to execute member functions of objects from different classes.

A pointer to a base class can point to objects of a derived class.

A.TRUE

B.FALSE A pure virtual function is a virtual function that

A.causes its class to be abstract.

B.returns nothing.

C.is used in a base class.

D.A and C

An abstract class is useful when

A.no classes should be derived from it.

B.there are multiple paths from one derived class to another.

C.no objects should be instantiated from it.

D.you want to defer the declaration of the class.

A friendfunction can access a class’s private data without being a member of the class.

A.TRUE

B.FALSE

A friend function can be used to

A.mediate arguments between classes.

B.increase the versatility of an overloaded operator.

C.allow access to an unrelated class.

D.B and C

The keyword friend appears in

A.the class allowing access to another class.

B.the private section of a class.

C.the public section of a class.

D.All of the above

A static function

A.should be called when an object is destroyed.

B.is closely connected to an individual object of a class.

C.can be called using the class name and function name.

D.is used when a dummy object must be created.

The user must always define the operation of the copy constructor.

A.TRUE

B.FALSE

The operation of the assignment operator and that of the copy constructor are

A.similar, except that the copy constructor creates a new object.

B.different, except that they both copy member data.

C.different, except that they both create a new object.

D.A and B

A copy constructor could be defined to copy only part of an object’s data.

A.TRUE

B.FALSE

The lifetime of a variable that is

A.local to a member function coincides with the lifetime of the function.

B.global coincides with the lifetime of a class.

C.nonstatic member data of an object coincides with the lifetime of the object.

D.A and C

There is no problem with returning the value of a variable defined as local within a member function so long as it is returned by value.

A.TRUE

B.FALSE

A copy constructor is invoked when

A.a function returns by value.

B.an argument is passed by value.

C.A and B

D.an argument is passed by reference.

What does the thispointer point to?

A.Data member of the class

B.the object of which the function using it is a member

C.Member function

D.Base class

A pointer is

A.the address of a variable.

B.an indication of the variable to be accessed next.

C.a variable for storing addresses.

D.the data type of an address variable.

The expression \*testcan be said to

A.refer to the contents of test.

B.dereference test.

C.refer to the value of the variable pointed to by test.

D.All of the above

A pointer to void can hold pointers to \_\_\_\_\_\_\_\_\_\_

A.int

B.float

C.char

D.Any data type

The type of variable a pointer points to must be part of the pointer’s definition so that

A.Data types don’t get mixed up when arithmetic is performed on them.

B.pointers can be added to one another to access structure members.

C.the compiler can perform arithmetic correctly to access array elements.

D.A and C

The first element in a string is

A.the name of the string.

B.the first character in the string.

C.the length of the string.

D.the name of the array holding the string.

The newoperator

A.returns a pointer to a variable.

B.creates a variable called new.

C.obtains memory for a new variable.

D.A and C

Definition for an array arrof 8 pointers that point to variables of type floatis

A.\*float arr[8]

B.float\* arr[8];

C.float pointer[8]

D.int \*ary[8]

The delete operator returns \_\_\_\_\_\_\_\_\_\_\_\_ to the operating system.

A.Memory that is no longer needed

B.Pointer

C.Object

D.Class In a linked list

A.each link contains a pointer to the next link.

B.each link contains data or a pointer to data.

C.the links are stored in an array.

D.A and B

If you wanted to sort many large objects or structures, it would be most efficient to

A.place them in an array and sort the array.

B.place pointers to them in an array and sort the array.

C.place them in a linked list and sort the linked list.

D.place references to them in an array and sort the array.

The contents of two pointers that point to adjacent variables of type floatdiffer by \_\_\_\_\_

A.1 byte

B.2 bytes

C.3 bytes

D.4 bytes

Which of the following is true about virtual functions in C++.

A.Virtual functions are functions that can be overridden in derived class with the same signature.

B.Virtual functions enable run-time polymorphism in a inheritance hierarchy.

C.If a function is 'virtual'in the base class, the most-derived class's implementation of the function is called according to the actual type of the object referred to, regardless of the declared type of the pointer or reference. In non-virtual functions, the functions are called according to the type of reference or pointer

D.All of the above

Predict the output of following C++ program.

#include<iostream>

using namespace std;

class Base {

public:

Base() { cout<<"Constructor: Base"<<endl; }

virtual ~Base() { cout<<"Destructor : Base"<<endl; }

};

class Derived: public Base {

public:

Derived() { cout<<"Constructor: Derived"<<endl; }

~Derived() { cout<<"Destructor : Derived"<<endl; }

};

int main() {

Base \*Var = new Derived();

delete Var;

return 0;

}

A.Constructor: Base

Constructor: Derived

Destructor : Derived

Destructor : Base

Predict the output of following C++ program. Assume that there is no alignment and a typical implementation of virtual functions is done by the compiler.

#include <iostream>

using namespace std;

class A

{

public:

virtual void fun();

};

class B

{

public:

void fun();

};

int main()

{int a = sizeof(A), b = sizeof(B);

if (a == b) cout <<"a == b";

else if (a >b) cout <<"a >b";

else cout <<"a <b";

return 0;

}

A.a>b

B.a==b

C.a<b

D.Compiler error

Which of the following is FALSE about references in C++

A.A reference must be initialized when declared

B.Once a reference is created, it cannot be later made to reference another object; it cannot be reset

C.References cannot be NULL

D.References cannot refer to constant value

#include <iostream>

using namespace std;

class A

{public:

virtual void fun() { cout <<"A::fun() "; }

};

class B: public A

{

public:

void fun() { cout <<"B::fun() "; }

};

class C: public B

{

public:

void fun() { cout <<"C::fun() "; }

};

iNt main()

{

B \*bp = new C;

bp->fun();

return 0;

}

Which function will be called by statements bp->fun();?

AA::fun()

B.B::fun()

C.C::fun()

D.Compiler error Which of the followings is/are automatically added to every class, if we do not write our own.

A.Copy Constructor

B.Assignment Operator

C.A constructor without any parameter

D.All of the above

What is the output of following program?

#include<iostream>

using namespace std;

class Point {

Point() { cout <<"Constructor called"; }

};

int main()

{Point t1;

return 0;

}

A.Compiler Error

B.Runtime Error

C.Constructor called

D.Segmentation Fault

What will be the output of following program?

#include <iostream>

using namespace std;

class Test

{public:

Test() { cout <<"Hello from Test() "; }

} a;

int main()

{

cout <<"Main Started ";

return 0;

}

AMain Started

B.Main Started Hello from Test()

C.Hello from Test() Main Started

D.Compiler Error: Global objects are not allowed

Which rule will not affect the friend function

A.private &protected members of a class cannot be accessed from outside

B.private &protected member can be accessed anywhere

C.both a &b

D.none of these

what is syntax of friend function?

A.Friend class1 Class2;

B.Friend class;

C.Friend class

D.none of these

what is output of the program?

#include<iostream>

using namespace std;

class Box

{double width;

public:

friend void printWidth(Box box);

void setWidth(double wid);};

void Box::setWidth(double wid)

{width-=wid}

void printWidth(Box box)

{box.width=box.width\*2;

cout<<"Width of box :"<<box.width<<endl}

int main()

{Box box;

box.setWidth(10.0);

printWidth(box);

return 0;}

A.40

B.5

C.10

D.20

pick out the correct statement.

A.A friend function may be member of another class

B.A friend function may not be member of another class

C.A friend function may or may not be member of another class

D.none of these

Where does keyword 'friend' should be placed?

A.Function declaration

B.Function definition

C.Main function

D.none of these

Which of the following is not type of constructor?

A.Copy constructor

B.Friend constructor

C.Default constructor

D.Parameterized constructor

Which of the following statement is correct?

A.Base class pointer cannot point to derived class

B.Derived class pointer cannot point to base class

C.Pointer to derived class cannot be created

D.Pointer to base class cannot be created

Which of the following is not the member of class?

A.Static function

B.Friend function

C.Const function

D.Virtual function

The operator used for dereferencing or indirection is\_\_\_\_\_\_\_\_\_\_\_\_\_\_

A.\*

B.&

C.->

D.->>

Choose the right option

string\* x, y

A.x is pointer to string, y is a string

B.y is pointer to string , x is a string

C.both x &y are pointer to string types

D.none of these

Which one of the following is not a possible state for a pointer?

A.hold the address of specific object

B.point one past the end of an object

C.Zero

D.point to tye

Which of the following is illegal?

A.int \*ip;

B.string s, \*sp=0;

C.int i;double \*dp=&i;

D.int \*pi=0;

what will happen in the code?

int a=100,b=200;

int \*p=&a, \*q=&b;

p=q;

A.b is assigned to a

B.p now points to b

C.a is assigned to b

D.q now points to a

what is output of this program?

#include<iostream>

using namespace std;

int main()

{int a=5, b=10, c=15;

int \*arr[]= {&a, &b, &c};

cout<<arr[1];

return 0;}

A.5

B.10

C.15

D.it will return some random number The correct statement for a function that takes pointer to a float , a pointer to a ponter to a char &return a pointer to a integer is

A.int\*\*fun(float\*\*, char\*\*)

B.int \*fun(float\*, char\*)

C.int \*\*\*fun(float\*, char\*\*)

D.int \*\*\*fun(\*float, \*\*char)

What is size of generic pointer in C++(in 32-bit platform)?

A.2

B.4

C.8

D.0

What is the output of this program?

#include<iostream>

using namespace std;

int main()

{int a[2][4]={3,6,9,12,15,18,21,24};

cout<<\*(a[1] + 2)<<\*(\*(a+1)+2)<<2[1[a]];

return 0;}

A.15 18 21

B.21 21 21

C.24 24 24

D.compile time error

Void pointer can point to which type of objects?

A.Int

B.Float

C.Double

D.all of above

When does the void pointer can be dereferenced?

A.when it doesn't point to any value

B.when it cast to another type of object

C.using delete keyword

D.none of above

The pointer can point to any variable that is not declared with which of these?

A.Const

B.Volatile

C.both a &b

D.Static

A void pointer can not point to which of these?

A.methods in C++

B.class member in c++

C.both a &b

D.none of these

To which does the function pointer point to?

A.Variable

B.Constants

C.Function

D.absolute variables

What we will not do with function pointers?

A.Allocation of memory

B.De-allocation of memory

C.both a &b

D.none of these

Which of the following can be passed in function pointers?

A.Variables

B.data types

C.Functions

D.none of these

Which operators are used in free store?

A.New

B.Delete

C.both a &b

D.none of these

What type of class member is operator new?

A.Static

B.Dynamic

C.Const

D.Smart

linked lists are not suitable to for the implementation of\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

A.insertion sort

B.radix sort

C.polynomial manipulation

D.binary search

Run time polymorphism can be achieved with\_\_\_\_\_\_\_\_\_\_\_\_

A.virtual base class

B.container class

C.virtual function

D.a &c

When a virtual function is redefine by the derived class, it is called\_\_\_\_\_\_

A.Overloading

B.Overriding

C.Rewriting

D.all of the above

An abstract class is useful when

A.no classes should be derived from it.

B.There are multiple paths from one derived class to another.

C.no objects should be instantiated from it.

D.you want to defer the declaration of the class.

Use of virtual functions implies

A.Overloading

B.Overriding

C.Static binding

D.Dynamic binding

Which of the following type casts will convert an Integer variable named amount to a Double type?

A.(double) amount

B.( int to double) amount

C. int to double(amount)

D.int (amount) to double

Pure virtual functions

A.Have to be redefined in the inherited class

B.Cannot have public access specification

C.Are mandatory for a virtual class

D.None of the above

A friend function to a class, C cannot access

A.Private data members and member functions

B.Public data members and member functions

C.Protected data members and member functions

D.The data members of the derived class of C

The function whose prototype is void getData(Item \*thing); receives

A.a pointer to a structure

B.a reference to a structure

C.a copy of a structure

D.None

The keyword friend does not appear in

A.The class allowing access to another class

B.The class desiring access to another class

C.The private section of a class

D.The public section of a class

What is the output of the following code

char symbol[3]={‘a’,‘b’,‘c’};

for (int index=0; index<3; index++)

cout <<symbol [index];

A.a b c

B.“abc”

C.abc

D.‘abc’

Predict output of the following program

#include<iostream>

using namespace std;

class Base

{public:

virtual void show() { cout<<" In Base \n"; }

};

class Derived: public Base

{public:

void show() { cout<<"In Derived \n"; }

};

int main(void)

{Base \*bp = new Derived;

bp->show();

Base &br = \*bp;

br.show();

return 0;

}

A.In Base

In Base

B.In Base

In Derived

C.In Derived

In Derived

D.In Derived

In Base

Output of following program

#include<iostream>

using namespace std;

class Base

{public:

virtual void show() { cout<<" In Base \n"; }

};

class Derived: public Base

{public:

void show() { cout<<"In Derived \n"; }

};

int main(void)

{Base \*bp, b;

Derived d;

bp = &d;

bp->show();

bp = &b;

bp->show();

return 0;

}

A.In Base

In Base

B.In Base

In Derived

C.In Derived

In Derived

D.In Derived

In Base

Which of the following is true about pure virtual functions?

1) Their implementation is not known in a class where they are declared.

2) If a class has a pure virtual function, then the class becomes abstract class and an instance of this class cannot be created.

A.Only 1

B.Only 2

C.Both

D.None

#include<iostream>

using namespace std;

class Base

{public:

virtual void show() = 0;

};

int main(void)

{Base b;

Base \*bp;

return 0;}

A.There are compiler errors in lines "Base b;" and "Base bp;"

B.There is compiler error in line "Base b;"

C.There is compiler error in line "Base bp;"

D.No compilation error

Predict the output of following program.

#include<iostream>

using namespace std;

class Base

{public:

virtual void show() = 0;

};

class Derived : public Base { };

int main(void)

{Derived q;

return 0;

}

A.Compiler Error: there cannot be an empty derived class

B.Compiler Error: Derived is abstract

C.No compiler Error

D.None

#include<iostream>

using namespace std;

class Base

{public:

virtual void show() = 0;

};

class Derived: public Base

{public:

void show() { cout<<"In Derived \n"; }

};

int main(void)

{

Derived d;

Base &br = d;

br.show();

return 0;

}

A.Compiler Error in line "Base &br = d;"

B.Empty output

C.In derived

D.None

Can a constructor be virtual? Will the following program compile?

#include <iostream>

using namespace std;

class Base {

public:

virtual Base() {}

};

int main() {

return 0;

}

AYes

BNo

Can a destructor be virtual? Will the following program compile?

#include <iostream>

using namespace std;

class Base {

public:

virtual ~Base() {}

};

int main() {

return 0;

}

A.Yes

B.No

Predict the output

#include<iostream>

using namespace std;

class Base {

public:

Base() { cout<<"Constructor: Base"<<endl; }

virtual ~Base() { cout<<"Destructor : Base"<<endl; }

};

class Derived: public Base {

public:

Derived() { cout<<"Constructor: Derived"<<endl; }

~Derived() { cout<<"Destructor : Derived"<<endl; }

};

int main() {

Base \*Var = new Derived();

delete Var;

return 0;

}

A.Constructor: Base

Constructor: Derived

Destructor : Derived

Destructor : Base

Can static functions be virtual? Will the following program compile?

#include<iostream>

using namespace std;

class Test

{public:

virtual static void fun() { }

};

AYes

B.No

Predict the output of following C++ program. Assume that there is no alignment and a typical implementation of virtual functions is done by the compiler.

#include <iostream>

using namespace std;

class A

{public:

virtual void fun();

};

class B

{public:

void fun();

};

int main()

{int a = sizeof(A), b = sizeof(B);

if (a == b) cout <<"a == b";

else if (a >b) cout <<"a >b";

else cout <<"a <b";

return 0;

}

A.a>b

B.a==b

C.a<b

D.Compile time error

#include <iostream>

using namespace std;

class A

{public:

virtual void fun() { cout <<"A::fun() "; }

};

class B: public A

{public:

void fun() { cout <<"B::fun() "; }

};

class C: public B

{public:

void fun() { cout <<"C::fun() "; }

};

int main()

{ B \*bp = new C;

bp->fun();

return 0;

}

A.a::fun()

B.b::fun()

C.c::fun()

D.None

Predict the output of following C++ program

#include<iostream>

using namespace std;

class Base

{public:

virtual void show() { cout<<" In Base \n"; }

};

class Derived: public Base

{public:

void show() { cout<<"In Derived \n"; }

};

int main(void)

{ Base \*bp = new Derived;

bp->Base::show(); // Note the use of scope resolution here

return 0;

}

A.In Base

B.In derived

C.Compile time error

D.Runtime error

Which of the following is true about this pointer?

A.It is passed as a hidden argument to all function calls

B.It is passed as a hidden argument to all non-static function calls

C.It is passed as a hidden argument to all static functions

D.None

What is the use of this pointer?

A.When local variable’s name is same as member’s name, we can access member using this pointer.

B.To return reference to the calling object

C.Can be used for chained function calls on an object

D.All

Predict the output of following C++ program.

#include<iostream>

using namespace std;

class Test

{private:

int x;

public:

Test(int x = 0) { this->x = x; }

void change(Test \*t) { this = t; }

void print() { cout <<"x = " <<x <<endl; }

};

int main()

{

Test obj(5);

Test \*ptr = new Test (10);

obj.change(ptr);

obj.print();

return 0;

}

A.X=5

B.X=10

C.Compile time error

D.Run time error

A static data member is given a value

A.Within the class definition

B.Outside the class definition

C.When the program is exeuted

D.Never

A function call mechanism that passes arguments to a function by passing a copy of the values of the arguments is \_\_\_\_\_\_\_\_\_\_

A.Call by name

B.Call by value

C.Call by reference

D.Call by value result

A ……………. takes a reference to an object of the same class as itself as an argument.

A.Reference constructor

B.Copy Constructor

C.Self Constructor

D.None of the above

Automatic initialization of object is carried out using a special member function called

A.Friend

B.Casting

C.Reference Parameter

D.Constructor

Which of the following condition is true for an object used as a function argument?

i) A copy of the entire objects is passed to the function.

ii) Only the address of the object is transferred to the function.

A.Only i

B.Only ii

C.Both I &ii

D.None

Which of the following parameter passing mechanism is/are supported by C++ not C

A.Pass by value

B.Pass by reference

C.Pass by value result

D.All of above

Every class has at least one constructor function, even when none is declared.

A.TRUE

B.FALSE

Can constructors be overloaded?

A.TRUE

B.FALSE

What is the difference between struct and class in terms of Access Modifier?

A.By default all the struct members are private while by default class members are public.

B.By default all the struct members are protected while by default class members are private.

C.By default all the struct members are public while by default class members are private.

D.By default all the struct members are public while by default class members are protected.

An abstract class can be instantiated.

A.TRUE

B.FALSE

The default access level assigned to members of a class is \_\_\_\_\_\_

A.Private

B.Public

C.Protected

D.Needs to be assigned

There is nothing like a virtual constructor of a class.

A.TRUE

B.FALSE

Which of the following operators allow defining the member functions of a class outside the class?

A.::

B.?

C.:?

D.%

Which type of class has only one unique value for all the objects of that same class?

A.This

B.Friend

C.Static

D.Both a and b

Which one of the following is not a fundamental data type in C++?

A.Float

B.string

C.Int

D.Char

What is a constructor?

A.A class automatically called whenever a new object of this class is created.

B.A class automatically called whenever a new object of this class is destroyed.

C.A function automatically called whenever a new object of this class is created.

D.A function automatically called whenever a new object of this class is destroyed.

Under what conditions a destructor destroys an object?

A.Scope of existence has finished

B.Object dynamically assigned and it is released using the operator delete.

C.Program terminated.

D.Both a and b.

If a member needs to have unique value for all the objects of that same class, declare the member as

A.Global variable outside class

B.Local variable inside constructor

C.Static variable inside class

D.Dynamic variable inside class

If a member needs to have unique value for all the objects of that same class, declare the member as

A.Global variable outside class

B.Local variable inside constructor

C.Static variable inside class

D.Dynamic variable inside class

When class B is inherited from class A, what is the order in which the constructers of those classes are called

A.Class A first Class B next

B.Class B first Class A next

C.Class B's only as it is the child class

D.Class A's only as it is the parent class

Which one of the following is not a valid reserved keyword in C++?

A.Explicit

B.Public

C.Implicit

D.Private

Variables declared in the body of a particular member function are known as data members and can be used in all member functions of the class.

A.TRUE

B.FALSE

In a class definition, data or functions designated private are accessible

A.to any function in the program.

B.only if you know the password.

C.to member functions of that class.

D.only to public members of the class.

A member function can always access the data

A.in the object of which it is a member.

B.in the class of which it is a member.

C.in any object of the class of which it is a member.

D.in the public part of its class.

Classes are useful because they

A.can closely model objects in the real world.

B.permit data to be hidden from other classes.

C.bring together all aspects of an entity in one place.

D.Options A, B and C

For the object for which it was called, a const member function

A.can modify both const and non-const member data.

B.can modify only const member data.

C.can modify only non-const member data.

D.can modify neither const nor non-const member data.

Dividing a program into functions

A.is the key to object-oriented programming.

B.makes the program easier to conceptualize.

C.may reduce the size of the program.

D.Option B and C

An expression

A.usually evaluates to a numerical value.

B.may be part of a statement.

C.always occurs outside a function.

D.Option A and B

A variable of type char can hold the value 301.

A.TRUE

B.FALSE

In an assignment statement, the value on the left of the equal sign is always equal to the value on the right.

A.TRUE

B.FALSE

It’s perfectly all right to use variables of different data types in the same arithmetic expression.

ATRUE

BFALSE

A function’s single most important role is to

Agive a name to a block of code.

Breduce program size.

Caccept arguments and provide a return value.

Dhelp organize a program into conceptual units.

A function argument is

A.a variable in the function that receives a value from the calling program.

B.a way that functions resist accepting the calling program’s values.

C.a value sent to the function by the calling program.

D.a value returned by the function to the calling program.

When arguments are passed by value, the function works with the original arguments in the calling program.

A.TRUE

B.FALSE

Which of the following can legitimately be passed to a function?

A.A constant

B.A variable

C.A structure

D.All of the above

How many values can be returned from a function?

A.0

B.1

When a function returns a value, the entire function call can appear on the right side of the equal sign and be assigned to another variable.

A.TRUE

B.FALSE

When an argument is passed by reference

A.a variable is created in the function to hold the argument’s value.

B.the function cannot access the argument’s value.

C.a temporary variable is created in the calling program to hold the argument’s value.

D.the function accesses the argument’s original value in the calling program.

Overloaded functions

A.are a group of functions with the same name.

B.all have the same number and types of arguments.

C.make life simpler for programmers.

D.A and C

A default argument has a value that

A.may be supplied by the calling program.

B.may be supplied by the function.

C.must have a constant value.

D.A and B

A static local variable is used to

A.make a variable visible to several functions.

B.make a variable visible to only one function.

C.retain a value when a function is not executing.

D.B and C

In C++ there can be an array of four dimensions.

A.TRUE

B.FALSE

When an array name is passed to a function, the function

A.accesses exactly the same array as the calling program.

B.refers to the array using a different name than that used by the calling program.

C.refers to the array using the same name as that used by the calling program.

D.A and B

The compiler will complain if you try to access array element 14 in a 10-element array.

A.TRUE

B.FALSE

The extraction operator (>>) stops reading a string when it encounters a space.

A.TRUE

B.FALSE

You can read input that consists of multiple lines of text using

A.the normal cout <<combination.

B.the cin.get() function with one argument.

C.the cin.get() function with two arguments.

D.the cin.get() function with three arguments.

You should prefer C-strings to the Standard C++ string class in new programs.

A.TRUE

B.FALSE

Objects of the string class

A.are zero-terminated.

B.can be copied with the assignment operator.

C.do not require memory management.

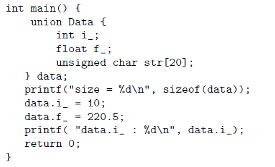
D.Both B and C

Can destuctors be private in C++?

A.Yes

B.No

What is value of size?



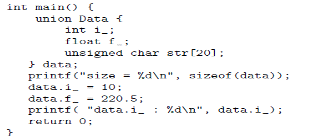
A.28

B.32

C.20

D.24

what value will be printed for data.?

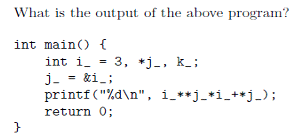


A.10 220.5 230.5 Unpredictable Value

B.220

C.230.5

D.Unpredictable Value

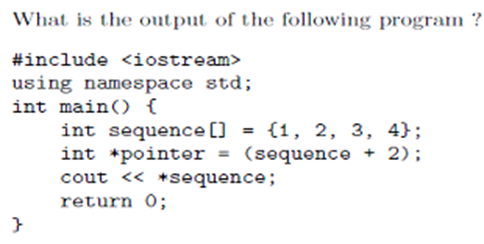


A25

B30

C9

D3



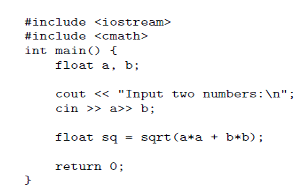
A.1

B.3

C.4

D.6

What is the compilation error for this program?

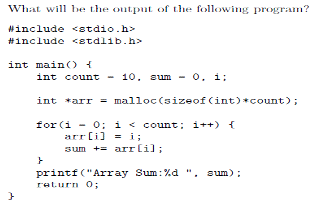


A.Each undeclared identifier is reported only once

B.cout and cin not declared in scope

C.invalid conversion from int to float

D.All of the above

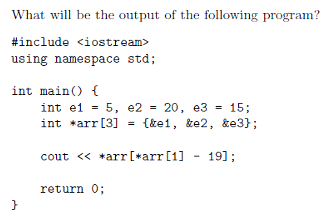


A.45

B.55

C.Array Sum: 45

D.Will not compile

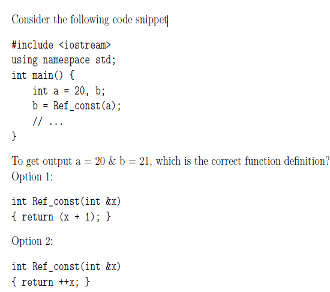


A.5

B.20

C.15

D.Unpredictable value

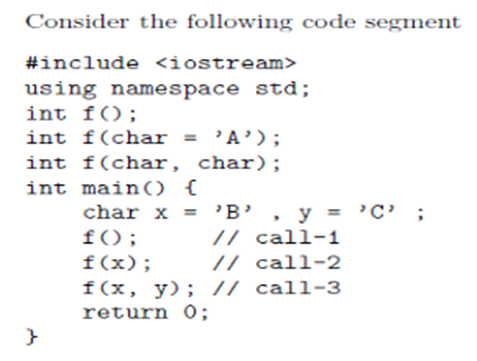


A.only 2

B.only 1

C.both 1 &2

D.None of the above

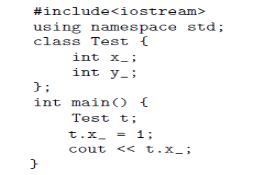


A.call-2

B.call-1

C.call-3

D.call-1, call-2



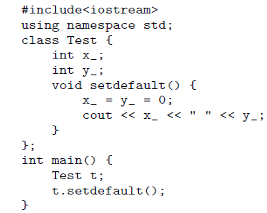
What will be the output of the program?

A.1

B.Default value

C.Will not compile

D.None of the above



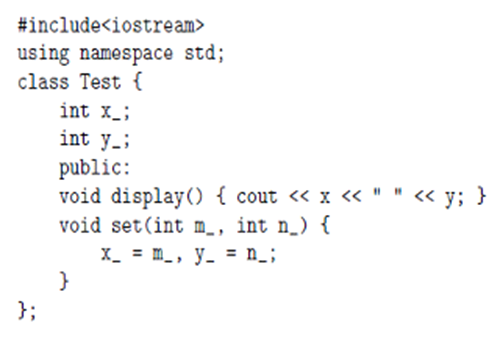
What is the output of the program?

A 0 0

B x = 0 y = 0

C 0

D Compilation error



Which function will change the state of the object?

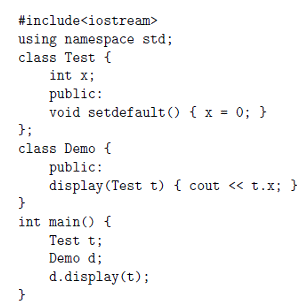
A Only set()

B Only display()

C display() and set() both

D None of the above

What will be the output of the following program?



A Compilation Error: display() cannot be accessed in application

B Compilation Error:Test class object cannot be accessed in function Demo

C Compilation Error: Variable x is private in Test

D Both A and B

The only that can be assigned directly to a pointer is \_\_\_\_\_\_\_\_\_

A.0

B.-1

C.999

D.-999

Which of the following feature is not supported by C++?

A.Exception Handling

B.Reflection

C.Operator Overloading

D.Namespace

Suppose aand bare integer variables and we form the sum a + b. Now suppose cand

dare floating-point variables and we form the sum c + d.The two +operators here are

clearly being used for different purposes. This is an example of \_\_\_\_\_\_\_\_\_\_\_\_

A.Operator Overloading

B.Inheritance

C.Function Overloading

D.Constructor

The operators that cannot be overloaded is

A.\*

B.-

C.::

D.()

Empty parentheses following a function name in a function prototype indicate that the function does not require any parameters to perform its task.

A.TRUE

B.FALSE

C++ programmers concentrate on creating , which contain data members and the member functions that manipulate those data members and provide services to clients.

A.Structures

B.Classes

C.Objects

D.Function

Which of the following is FALSE about references in C++

A.A reference must be initialized when declared

B.Once a reference is created, it cannot be later made to reference another object; it cannot be reset

C.References cannot be NULL

D.References cannot refer to constant value

What will be the output of following program?

#include <iostream>

using namespace std;

class Test

{public:

Test() { cout <<"Hello from Test() "; }

} a;

int main()

{cout <<"Main Started ";

return 0;

}

A.Main Started

B.Main Started Hello from Test()

C.Hello from Test() Main Started

D.Compiler Error: Global objects are not allowed

Which of the following is true about constructors.

•They cannot be virtual.

•They cannot be private.

•They are automatically called by new operator

A.All 1, 2, and 3

B.Only 1 and 3

C.Only 1 and 2

D.Only 2 and 3

Which of the following operators are overloaded by default by the compiler?

1) Comparison Operator ( == )

2) Assignment Operator ( = )

A.Both 1 and 2

BOnly 1

C.Only 2

D.None of the two

Which of the following is true about inline functions and macros.

A.Inline functions do type checking forparameters, macros don't

B.Macros cannot have return statement, inline functions can

C.Macros are processed by pre-processor and inline functions are processed in later stages of compilation.

D.All of the above

In C++, const qualifier can be applied to

•Member functions of a class

•Function arguments

•To a class data member which is declared as static

•Reference variables

A.Only 1, 2 and 3

B.Only 1, 2 and 4

CAll

DOnly 1, 3 and 4

In C++ ..................... operator is used for Dynamic memory allocation.

A.Scope resolution

B.Conditional

C.New

D.Membership access

What is the output of the program

#include<iostream.h>

void main()

{

int n=1;

cout<<endl<<"The numbers are;"<<endl;

do

{cout <<n<<"\t";

n++;

} while (n<=100);

cout <<endl;

}

APrint natural numbers 0 to 99

B Print natural numbers 1 to 99

C Print natural numbers 0 to 100

D Print natural numbers 1 to 100

Answer D

Marks 2 M

Unit I

Id 71

Question Because the lifetime of a local variable is limited and determined automatically, these variables are also called

A Automator

B Automatic

C Dynamic

D Static

Answer B

Marks 1 M

Unit I

Id 72

Question Which of the following header file includes definition of cin and cout?

A istream.h

B ostream.h

C iomanip.h

D iostream.h

Answer D

Marks 1

Unit I

Id 73

Question Which of the following statements regarding inline functions is correct?

A It speeds up execution

B It slows down execution

C It increases the code size

D Both A and C.

Answer D

Marks 1

Unit I

Id 74

Question Which of the following access specifier is used as a default in a class definition?

A Public

B Private

C Protected

D Friend

Answer B

Marks 1

Unit I

Id 75

Question Which of the following statements is correct in C++?

A Classes cannot have data as protected members.

B Structures can have functions as members.

C Class members are public by default.

D Structure members are private by default.

Answer B

Marks 1

Unit I

Id 76

Question cout is a/an \_\_\_\_\_\_\_\_\_\_

A operator

B Function

C object

D Macro

Answer C

Marks 1

Unit I

Id 77

Question Which of the following concepts of OOPS means exposing only necessary information to client?

A Encapsulation

B Abstraction

C Data hiding

D Data binding

Answer C

Marks 1

Unit 1

Id 78

Question Which of the following keywords is used to control access to a class member?

A Default

B Break

C Protected

D Asm

Answer C

Marks 1

Unit 1

Id 79

Question Utility functions are also called as .......

A Virtual function

B Friend function

C Helper function

D None of above

Answer C

Marks 1

Unit 1

Id 80

Question ......... is a member function with the same name as the class.

A Friend function

B Constructor

C Destructor

D None of above

Answer B

Marks 1

Unit 1

Id 81

Question Which is not the feature of constructor?

A It cannot be inherited.

B It should be declared in Private.

C It do not have return type

D All of above

Answer B

Marks 1

Unit 1

Id 82

Question Which is not type of constructor?

A Default

B Copy

C Parameterized

D None of above

Answer D

Marks 1

Unit 1

Id 83

Question Objects are destroyed in the reverse order of its creation.

A True

B False

C -

D -

Answer A

Marks 1

Unit 1

Id 84

Question .... constructor is used for copying the object of same class type.

A Copy

B Default

C Parameterized

D None of above

Answer A

Marks 1

Unit 1

Id 85

Question The function inside a class is called as .......

A Class Function

B Member Function

C Method

D All of above

Answer B

Marks 1

Unit 1

Id 86

Question Which operator is used to define member function of a class outside the class?

A !

B :

C ::

D .

Answer C

Marks 1

Unit 1

Id 87

Question How many objects can be created by a class?

A 1

B 2

C 3

D As Many as required

Answer D

Marks 1

Unit 1

Id 88

Question Default return type of C++ main( ) is .....

A float

B void

C Int

D Pointer

Answer C

Marks 1

Unit 1

Id 89

Question Enumerated data type is ........

A User-defined data type

B In-built data type

C Derived data type

D None of above

Answer A

Marks 1

Unit 1

Id 90

Question Attributes of a class are called as ........

A Member functions

B Data members

C Objects

D All of above

Answer B

Marks 1

Unit 1

Id 91

Question Class acquire space in memory.

A True

B False

C -

D -

Answer B

Marks 1

Unit 1

Id 92

Question In object-oriented programming ........ is more important.

A Function

B Procedure

C Data

D All of above

Answer C

Marks 1

Unit 1

Id 93

Question Object-oriented programming follows Top-down approach.

A True

B False

C -

D -

Answer B

Marks 1

Unit 1

Id 94

Question The following operators can not be overloaded

A Unary operator

B Binary operator

C Ternary operator

D None of the above

Answer C

Marks 1

Unit 1

Id 95

Question C++ does not supports the following

A Multilevel inheritance

B Hierarchical inheritance

C Hybrid inheritance

D None of the above

Answer D

Marks 1

Unit 1

Id 96

Question Which of the following is not the keyword in C++?

A Volatile

B Friend

C Extends

D This

Answer C

Marks 1

Unit 1

Id 97

Question Which data type can be used to hold a wide character in C++?

A unsigned char;

B Int

C wchar\_t

D None of the above

Answer C

Marks 1

Unit 1

Id 98

Question Which type is best suited to represent the logical values?

A integer

B boolean

C character

D all of the mentioned

Answer B

Marks 1

Unit 1

Id 99

Question The following is the C++ style comment

A //

B /\*..\*/

C –

D None of above

Answer A

Marks 1

Unit 1

Id 100

Question Which of the following statements is false?

A Every C++ program must have a main().

B In C++, white spaces and carriage returns are ignored by the compiler.

C C++ statements terminate with semicolon.

D Main() terminates with semicolon.

Answer D

Marks 1

Unit 1

Id 101

Question Which of the following statements regarding comments is false?

A /\*..\*/

B Comment beginning with // extends to the end of the line

C Comments may be nested

D Comments are used to describe a program

Answer C

Marks 1

Unit 1

Id 102

Question The result of the following statement is

int y = 7;

int ans = ++y;

cout<<”ans=”<<ans;

cout<<”y”<<y;

A ans=7, y=7

B ans=8,y=7

C ans=8,y=8;

D None

Answer C

Marks 1

Unit 1

Id 103

Question Inline functions are

A Declared in the class defined outside the class

B Defined outside the class using keyword inline

C Defined inside the class using keyword inline

D None of the above

Answer B

Marks 1

Unit 1

Id 104

Question Functions can returns

A Arrays

B References

C Objects

D All of above

Answer D

Marks 1

Unit 1

Id 105

Question Which of the following control expressions are valid for an if statement?

A an integer expression

B a Boolean expression

C either A or B

D Neither A nor B

Answer C

Marks 1

Unit 1

Id 106

Question State true of false.

i) We cannot make the function inline by defining a function outside the class.

ii) A member function can be called by using its name inside another member function of the same class, this is known as nesting of member function.

A True, True

B True, False

C False, True

D False, False

Answer C

Marks 1

Unit 1

Id 107

Question What will be the values of x, m and n after execution of the following statements?

Int x, m, n;

m=10;

n=15;

x= ++m + n++;

A x=25, m=10, n=15

B x=27, m=10, n=15

C x=26, m=11, n=16

D x=27, m=11, n=16

Answer C

Marks 2

Unit 1

Id 108

Question How many types of polymorphisms are supported by C++?

A 1

B 2

C 3

D 4

Answer B

Marks 1

Unit 1

Id 109

Question Which of the following approach is adapted by C++?

A Top-down

B Bottom-up

C Right-left

D Left-right

Answer B

Marks 1

Unit 1

Id 110

Question Which of the following is the correct class of the object cout?

A Iostream

B istream

C Ostream

D Ifstream

Answer C

Marks 1

Unit 1

Id 111

Question Which of the following functions are performed by a constructor?

A Construct a new class

B Construct a new object

C Construct a new function

D Initialize objects

Answer D

Marks 1

Unit 1

Id 112

Question Which of the following ways are legal to access a class data member using this pointer?

A this->x

B this.x

C \*this.x

D \*this-x

Answer A

Marks 1

Unit 1

Id 113

Question Which operator is having right to left associativity in the following?

A Array subscripting

B Function call

C Addition and subtraction

D Type cast

Answer D

Marks 1

Unit 1

Id 114

Question Which operator is having the highest precedence?

A Postfix

B Unary

C Shift

D Equality

Answer A

Marks 1

Unit 1

Id 115

Question #include <iostream>

using namespace std;

int main()

{

int a;

a = 5 + 3 \* 5;

cout <<a;

return 0;

}

A 35

B 20

C 25

D 30

Answer B

Marks 2

Unit 1

Id 116

Question #include <iostream>

using namespace std;

main()

{

double a = 21.09399;

float b = 10.20;

int c ,d;

c = (int) a;

d = (int) b;

cout <<c <<' '<<d;

return 0;

}

A 20 10

B 10 21

C 21 10

D None

Answer A

Marks 2

Unit 1

Id 117

Question #include <iostream>

using namespace std;

int main()

{

int num1 = 10;

float num2 = 20;

cout <<sizeof(num1 + num2);

return 0;

}

A 2

B 4

C 8

D Garbage

Answer B

Marks 2

Unit 1

Id 118

Question #include <stdio.h>

using namespace std;

int array1[] = {1200, 200, 2300, 1230, 1543};

int array2[] = {12, 14, 16, 18, 20};

int temp, result = 0;

int main()

{

for (temp = 0; temp <5; temp++) {

result += array1[temp];

}

for (temp = 0; temp <4; temp++) {

result += array2[temp];

}

cout <<result;

return 0;

}

A 6553

B 6533

C 6522

D 12200

Answer B

Marks 2

Unit 1

Id 119

Question In procedural programming the focus in on …...........

A data

B structure

C function

D pointers

Answer C

Marks 1

Unit 1

Id 120

Question In object oriented programming the focus is on ….......

A data

B structure

C function

D pointers

Answer A

Marks 1

Unit 1

Id 121

Question Which of the following feature of procedure oriented program is false?

A Makes use of bottom up approach

B Functions share global data

C The most fundamental unit of program is function

D All of these

Answer A

Marks 1

Unit 1

Id 122

Question Which of the following feature of object oriented program is false?

A Data and Functions can be added easily

B Data can be hidden from outside world

C Object can communicate with each other

D The focus is on procedures

Answer D

Marks 1

Unit 1

Id 123

Question C++ was originally developed by ….......

A Donald Knuth

B Bjarne Sroustrups

C Dennis Ritchie

D None of these

Answer B

Marks 1

Unit 1

Id 124

Question Which of the following approach is adopted in C++?

A Top down

B Bottom up

C Horizontal

D Vertical

Answer B

Marks 1

Unit 1

Id 125

Question Which feature of C++ contain the concept of super class and subclass?

A Class and object

B Encapsulation

C Abstraction

D Inheritance

Answer D

Marks 1

Unit 1

Id 126

Question The main intention of using inheritance is ….........

A to help in converting one data type to other

B to hide the details of base class

C to extend the capabilities of base class

D to help in modular programming

Answer C

Marks 1

Unit 1

Id 127

Question If particular software can be used in some other application than the one

for which it is created then it reveals ….........

A data binding

B data reusability

C data encapsulation

D none of these

Answer B

Marks 1

Unit 1

Id 128

Question Which of the following data type does not return anything?

A Int

B short

C long

D void

Answer D

Marks 1

Unit 1

Id 129

Question How many objects can be created from an abstract class?

A Zero

B One

C Two

D As many as we want

Answer A

Marks 1

Unit 1

Id 130

Question Which of the following statements is correct for a static member function?

1. It can access only other static members of its class.

• It can be called using the class name, instead of objects

A Only 1 is correct

B Only 2 is correct

C Both 1 and 2 are correct

D Both 1 and 2 are incorrect

Answer C

Marks 1

Unit 1

Id 131

Question Select the correct statement

I. In procedural programming oriented language all the function calls are

resolved at compile time.

II. In object oriented programming language all function calls are

resolved at compile time.

A Only I

B Only II

C Both I and II

D Neither I nor II

Answer A

Marks 1

Unit 1

Id 132

Question What happens when a class with parameterized constructors and having no default constructor is used in a program and we create an object that needs a zero-argument constructor?

A Compile-time error

B Preprocessing error

C Runtime error

D Runtime exception

Answer A

Marks 1

Unit 1

Id 133

Question Which of the following interface determines how your program will be used by other program?

A Public

B Private

C Protected

D None of these

Answer A

Marks 1

Unit I

Id 134

Question What is the difference between struct and class in C++?

A All members of a structure are public and structures don't have constructors and destructors

B Members of a class are private by default and members of struct are public by default. When deriving a struct from a class/struct, default access-specifier for a base class/struct is public and when deriving a class, default access specifier is private.

C All members of a structure are public and structures don't have virtual functions

D All above

Answer B

Marks 1

Unit I

Id 135

Question Predict the output of following C++ program

#include<iostream>

using namespace std;

class Empty {};

int main()

{

cout <<sizeof(Empty);

return 0;

}

A A non zero value

B 0

C Compile time error

D Runtime error

Answer A

Marks 1

Unit I

Id 136

Question class Test {

int x;

};

int main()

{

Test t;

cout <<t.x;

return 0;

}

A 0

B Garbage value

C Compile time error

D

Answer C

Marks 1

Unit I

Id 137

Question Which of the following is true?

A All objects of a class share all data members of class

B Objects of a class do not share non-static members. Every object has its own copy

C Objects of a class do not share codes of non-static methods, they have their own copy

D None

Answer B

Marks 1

Unit I

Id 138

Question Assume that an integer and a pointer each takes 4 bytes. Also, assume that there is no alignment in objects. Predict the output following program.

#include<iostream>

using namespace std;

class Test

{

static int x;

int \*ptr;

int y;

};

int main()

{

Test t;

cout <<sizeof(t) <<" ";

cout <<sizeof(Test \*);

}

A 12 4

B 12 12

C 8 4

D 8 8

Answer

Marks 2

Unit I

Id 139

Question Which of the following is true about the following program

#include <iostream>

class Test

{

public:

int i;

void get();

};

void Test::get()

{

std::cout <<"Enter the value of i: ";

std::cin >>i;

}

Test t; // Global object

int main()

{

Test t; // local object

t.get();

std::cout <<"value of i in local t: "<<t.i<<'\n';

::t.get();

std::cout <<"value of i in global t: "<<::t.i<<'\n';

return 0;

}

A Compiler Error: Cannot have two objects with same class name

B Compiler Error in Line "::t.get();"

C Compiles and runs fine

D

Answer C

Marks 2

Unit I

Id 140

Question How to create a dynamic array of pointers (to integers) of size 10 using new in C++? Hint: We can create a non-dynamic array using int \*arr[10]

A int \*arr = new int \*[10];

B int \*\*arr = new int \*[10];

C int \*arr = new int [10];

D Not possible

Answer 1

Marks 1

Unit I

Id 141

Question Which of the following is true about new when compared with malloc. 1) new is an operator, malloc is a function 2) new calls constructor, malloc doesn't 3) new returns appropriate pointer, malloc returns void \* and pointer needs to typecast to appropriate type.

A 1 and 3

B 2 and 3

C 1 and 2

D All 1,2,3

Answer C

Marks 1

Unit I

Id 142

Question Predict the output?

#include <iostream>

using namespace std;

class Test

{

int x;

Test() { x = 5;}

};

int main()

{

Test \*t = new Test;

cout <<t->x;

}

A Compile time error

B Garbage

C 0

D 5

Answer A

Marks 2

Unit I

Id 143

Question What happens when delete is used for a NULL pointer?

int \*ptr = NULL;

delete ptr;

A Compile time error

B Run time error

C No effect

D

Answer C

Marks 1

Unit I

Id 144

Question Is it fine to call delete twice for a pointer?

#include<iostream>

using namespace std;

int main()

{

int \*ptr = new int;

delete ptr;

delete ptr;

return 0;

}

A Yes

B No

C

D

Answer B

Marks 1

Unit I

Id 145

Question Which of the followings is/are automatically added to every class, if we do not write our own.

A Copy constructor

B Assignment operator

C A constructor without any parameter

D All

Answer D

Marks 1

Unit I

Id 146

Question When a copy constructor may be called?

A When an object of the class is returned by value

B When an object of the class is passed (to a function) by value as an argument

C When an object is constructed based on another object of the same class

D All

Answer D

Marks 1

Unit I

Id 147

Question Output of following program?

#include<iostream>

using namespace std;

class Point {

Point() { cout <<"Constructor called"; }

};

int main()

{

Point t1;

return 0;

}

A Compile time error

B Run time error

C Constructor called

D

Answer A

Marks 1

Unit I

Id 148

Question Which of the following interface determines how your program will be used by other program?

A Public

B Private

C Protected

D None of these

Answer A

Marks 1

Unit I

Id 149

Question #include<iostream>

using namespace std;

class Point {

public:

Point() { cout <<"Constructor called"; }

};

int main()

{

Point t1, \*t2;

return 0;

}

A Compiler Error

B Constructor called

Constructor called

C Constructor called

D

Answer C

Marks 1

Unit I

Question #include<iostream>

using namespace std;

class X

{

public:

int x;

};

int main()

{

X a = {10};

X b = a;

cout <<a.x <<" " <<b.x;

return 0;

}

A Compiler Error

B 10 followed by Garbage Value

C 10 10

D 10 0

Answer D

Marks

Question

POLYMORPHISM Polymorphism is supported by the c++ by using following ways

Afunction overloading

Boperator overloading

Cvirtual functions

Dall of the above

Compile time polymorphism is supported by

Afunction overloading

Bvirtual function

Coperator overloading

Dboth a&c

Question Run time polymorphism is supported by

A.function overloading

B.operator overloading

C.virtual function

D.both a and b

Selecting the appropriate overloaded function by the compiler is known as

A.late binding

B.early binding

C.both a and b

D.none of the above

object to function binding is done at compile time then is it known as

Aearly binding

Bcompile time binding

Cnone of the above

Dboth a and b

Question Operator overloading is ---.

Arun time polymorphism

Bcompile time polymorphism

Cnone of the above

Dboth a and b

Which of the following operator cannot be overloaded

A.scope resolution operator(::)

B.Size of operator (sizeof[])

C.Conditional operator(?:)

D.All of the above

While performing operator overloading which function/keyword we have to use

A.Function

B.Operator

C.Op

D.none of the above

Which of the statement is not true about operator overloading

Awe can overload only existing operator

Bbasic meaning cannot be changed

Cbinary operator should have return type

DAll of the above

Pick up the correct statement related with operator overloading

Awe can overload a class access operator

Bwe can change the meaning of basic operator

Cbinary operator should have a return type

Dboth a and b

We are overloading a unary operator without friend function how many argument we have to pass

A.1

B.2

C.0

D.none of the above

Suppose we are overloading a binary operator with friend function, how many parameter of argument we have to pass

A.1

B.2

C.3

D.none of ths above

we are overloading a binary operator without friend function how many argument we have to pass

A.1

B.2

C.0

D.none of the abve

What is polymorphism

A.it is ability to take many forms

B.it is instance of class

C.one class acquire the properties of another class

D.All of the above

What is true about the operator overloading

A.with friend function we need to pass two arguments for binary operator

B.with friend function we need to pass one arguments for unary operator

C.both a and b

D.none of the above

\_\_\_\_\_\_allows you to give special meaning to some operator when there are operands associated with it.

A.function overloading

B.virtual function

C.operator overloading

D.none of the above

Converting from small to larger data type is known as \_\_\_\_.

A.promotion

B.operator

C.polymorphism

D.none of the above

what are the types of type conversion

A.implicit

B.explicit

C.both a and b

D.none of the above

Reusability is supported by following feature

A.polymorphisms

B.message passing

C.inheritance

D.operator overloading

Deriving a new class from a base class is known as \_\_.

A.polymorphisms

B.inheritance

C.message passing

D.operator overloading

Base class is also known as\_\_\_.

A.super class

B.parent class

C.both a and b

D.none of the above

Child class is also known as

A.sub class

B.derived class

C.both a and b

D.known class

Derived class\_\_\_ cannot access from base class

A.constructor

B.destructors

C.copy constructor

D.all of the above we can derive a new class from a derived class

A.true

B.fals

How many parameter does a conversion operator take?

A.0

B.2

C.3

D.as many as possible

\_\_is used to define pure virtual function?

A.&

B.=0

C.@

D.\*

Which is also known as abstract class?

A.virtual function

B.pure virtual function

C.derived class function

D.base class function pick the correct option

A.We can make the instance of the abstract class

B.We can not make the instance of the abstract class

C.both a and b

D.none of the above

How many access specifiers are there in c++?

A.2

B.3

C.5

D.4 Where we have to use an abstract class?

A.in base class only

B.in derived class only

C.both and b

D.None of the above

For what we can apply access

A.function

B.data member

C.none of the above specifier

D.both a and c

What is default access specifier for class?

A.public

B.protected

C.private

D.none of the above

We have to define a constructor for the derived class must be required\_\_\_\_.

A.if base class constructor does not require arguments

B.if base/parent class constructor required arguments

C.no need

D.always

Use of the friend function is\_\_\_.

A.the class allowing access to another class

B.the private section of a class

C.the public section of the class

D.all of the above

If an attribute is private define then which method can have access to it

A.only static function

B.only functions of that class

C.only method in that package

D.none of the above What is syntax of deriving a new class from base class is\_\_\_.

A.class name, new class name

B.new class name, base class name

C.class name: access specifier class name

D.none of the above Which constructor will initialize the base class data member

A.base class

B.derived class

C.derived derived class

D.none of these

Inheritance can be done using :: symbol

A.True

B.False

When we derived a new class using more than one class then type of inheritance is known as\_\_\_\_.

A.multiple inheritance

B.single inheritance

C.hybrid inheritance

D.multilevel inheritance

When class B is derive from A , and class C is derived from B, this kind of inheritance is known as \_\_\_\_\_\_.

A.multiple inheritance

B.single inheritance

C.hybrid inheritance

D.multilevel inheritance

The base class will provide you\_\_\_\_\_\_\_\_\_\_.

A.specific objects than the derived class

B.more generalized version of derived class

C.empty template of base class

D.all of the above Pick up the correct statement form following

i)we have to use abstract keyword to define the abstract class

ii)inheritance allows multilevel class hierarchies more than two levels also

iii)reusability is supported by derivation

iv) we can change the meaning of operator during operator overloading

A.i only

B.ii only

C.i and ii only

D.ii and iii only

What does derived class does not inherit from the base class

i)constructor

ii)destructor

iii)operator=()

iv)friends

A.i and ii

B.ii and iii

C.only i, iii and iv

D.all of the above

Choose the correct the statement from following

i)inheritance supported in terms of single, multiple , multilevel ,hybrid inheritance

ii) polymorphisms is supported by function overloading , operator overloading and virtual function

iii) abstraction is not supported by c++

A.i is correct

B.only ii is correct

C.i and ii is correct

D.none of the above

\_\_\_\_\_types of classes in c++.

A.1

B.2

C.3

D.4

:\_\_\_\_\_\_\_is used to define a pure virtual function.

A.$

B.^

C.=0

D.#

Pick up the correct statement

A.a base class may have more than one class

B.derived class can be derived from more than one class

C.both a and b

D.none of the above

class A: public B, public C is a type of inheritance

A.Single

B.Multiple

C.Multilevel

D.Hybrid

When we have to use the mutable keyword

A.data member to change within a const member function

B.not allow the data member to change within a const member function

C.it will copy the values of the variable

D.none of the above mentioned

choose the correct statement

A.destructor cannot be inherited

B.private member not inherited to derived class

C.constructor cannot inherited

D.a and c

Use of function or operator to act different ways on different data type is called as

A.derivation

B.inheritance

C.polymorphisms

D.none of the above

Choose the correct statement

A.Constructor has a return type

B.constructor always define in public scope

C.constructor has same name that of class name

D.b and c

overloading of a prefix increment operator by means of a member function takes\_\_\_\_\_\_\_\_\_.

A.one argument

B.two argument

C.no argument

D.none of the above

Pick up the correct statement

i)abstract type of class should contain at least one virtual function

ii)we can create an object of abstract class

iii)abstract class is used to provide an interface to subclasses

iv)abstract class can also have normal function

A.i and ii only

B.i , ii, and iii only

C.i ,iii and iv

D.all of the above

What is function overloading

A.we have to use same function name but different parameter

B.different function name but same parameter

C.both a and b

D.none of the above

Virtual base class is used to \_\_\_\_.

A.to perform operator overloading

B.to perform function overloading

C.to remove ambiguity in multiple inheritance

D.all of the above

Pick up the correct statement

A.protected member from base class can be accessed by own class and its all subclasses

B.protected member are not inherited by any other class

C.Protected member are combination of public and private access member

D.all of the above

pick up the correct statement

A.base class and derived class can have their own constructor

B.base class and derived class can have their own destructor

C.neither a or neither b

D.both a and b

Make a correct sequence of a statement

i)destructor of derived class is called

ii)destructor of base class is called

iii)constructor of derived class is called

iv)constructor of base class is called

A.i,ii,iv,iii

B.iv,iii,ii,i

C.iv,iii,i,ii

D.i,ii,iii,iv

Operator overloading is

A.making C++ operators work with objects.

B.giving C++ operators more than they can handle.

C.giving new meanings to existing C++ operators.

D.Both A and C Assume a class C with objects obj1, obj2, and obj3. For the statement obj3 = obj1 - obj2 to work correctly, the overloaded - operator must

A.take two arguments.

B.return a value.

C.use the object of which it is a member as an operand.

D.Both B and C

When you overload an arithmetic assignment operator, the result

A.goes in the object to the right of the operator.

B.goes in the object to the left of the operator.

C.goes in the object of which the operator is a member.

D.Both B and C

To convert from a user-defined class to a basic type, you would most likely use

A.a built-in conversion operator.

B.a one-argument constructor.

C.an overloaded = operator.

D.a conversion operator that’s a member of the class.

An overloaded operator always requires one less argument than its number

of operands.

A.TRUE

B.FALSE

The compiler won’t object if you overload the \* operator to perform division.

A.TRUE

B.FALSE

Inheritance is a way to

A.make general classes into more specific classes.

B.pass arguments to objects of classes.

C.add features to existing classes without rewriting them.

D.A and C

Advantages of inheritance include

A.providing a useful conceptual framework.

B.facilitating class libraries.

C.avoiding the rewriting of code.

D.All of the above

Adding a derived class to a base class requires fundamental changes to the base class.

A.TRUE

B.FALSE

Id 71

Question To be accessed from a member function of the derived class, data or functions in the base class must be

A.public

B.private

C.protected

D.static

If a base class contains a member function basefunc(), and a derived class does not contain a function with this name, can an object of the derived class access basefunc()?

A.YES

B.NO

If no constructors are specified for a derived class, objects of the derived class will use the constructors in the base class.

A.TRUE

B.FALSE

The scope-resolution operator usually

A.specifies a particular class.

B.tells what base class a class is derived from.

C.resolves ambiguities.

D.A and C

Assume a class Derv that is privately derived from class Base. An object of class Derv located in main() can access

A.public members of Derv.

B.protected members of Derv.

C.private members of Derv.

D.public members of Base.

True or False: A class Dcan be derived from a class C, which is derived from a class B, which is derived from a class A.

A.TRUE

B.FALSE

It is illegal to make objects of one class members of another class.

A.TRUE

B.FALSE

A class hierarchy

A.shows the same relationships as an organization chart.

B.describes “has a” relationships.

C.describes “is a kind of” relationships.

D.shows the same relationships as a family tree.

What is the output of the program?

#include <iostream>

#include <string>

using namespace std;

class Department {

public:

string dept;

Department(string d):dept(d) { }

void getDeptName() { cout <<dept; }

};

class Student : private Department {

public:

string name;

Student(string n = "Not entered", string d = "ATDC") :

name(n), Department(d) { }

using Department::getDeptName;

};

int main() {

Student s("CSE");

s.getDeptName();

return 0;

}

A.CSE

B.ATDC

C.Not entered

D.Compilation Error

Identify the lines on which the compiler will report an error.

#include <iostream>// ---1

using namespace std; // ---2

class Base { // ---3

int var\_; // ---4

public: // ---5

Base():var\_(0){} // ---6

}; // ---7

class Derived: public Base { public: // ---8

int varD\_; // ---9

void print () { cout <<var\_; } // ---10

}; // ---11

int main() { // ---12

Derived d; // ---13

d.var\_ = 1; // ---14

d.varD\_ = 1; // ---15

cout <<d.var\_ <<""<<d.varD\_; // ---16

return 0; // ---17

} // ---18

A.6, 10, 14, 15

B.6, 15

C.6, 14, 16

D.10, 14, 16

#include <iostream>

using namespace std;

class Base { public:

int var\_;

void func(int){}

};

class Derived: public Base { public:

int varD\_;

void func(int){}

};

int main() {

Derived d;

d.func(1);

return 0;

}

Which of the following function will be invoked by d.func(1)?

A.Base::func(int)

B.Derived::func(int)

C.Compilation Error

D.None of the above

What is the output of the following program?

#include<iostream>

#include<string>

using namespace std;

class Base {

public:

void func\_f1(int i) { cout <<"In base func\_f1 "; }

void func\_f2(int i) { cout <<"In base func\_f2 "; }

};

class Derived: public Base {

public:

void func\_f1(int i ) { cout <<"In derived func\_f1 "; }

void func\_f1(string s) { cout <<"func\_f1 string "; }

void func\_f3(int i) { cout <<"In derived func\_f3 "; }

};

int main() {

Base b;

Derived d;

d.func\_f1(3);

d.func\_f1("Blue");

d.func\_f3(3);

d.func\_f2(3);

return 0;

}

A.Compilation Error: Cannot add new parameters to func\_f1

B.In derived func\_f1 func\_f1 string In derived func\_f3 In base func\_f2

C.In base func\_f2 func\_f1 string In derived func\_f3 In derived func\_f1

D.Compilation Error: Cannot define func\_f3 containing same parameter type as func\_f1

What is the output of the following program? {Assume size of int as 4}

#include<iostream>

using namespace std;

class base {

int data;

};

class derived1: public base { };

class derived2: public derived1 { };

int main() {

cout <<sizeof(derived2);

return 0;

}

A.4

B.8

C.12

D.16

What will be the output of the following program?

#include <iostream>

using namespace std;

class B{ public: int base;

B() {}

~B() {}

};

class D: public B { public: int derived;

D() {}

~D() {}

};

int main() {

D d1;

B b1;

cout <<&b1.base <<"";

cout <<&d1.base;

return 0;

}

A.0x28fef8 0x28fef8

B.0x28fef8 0x28fefc

C.Compilation Error

D.None of the above

What will be the output of the following program?

#include<iostream>

using namespace std;

class Base { public:

Base() { cout <<"Base Ctor"<<endl; }

~Base() { cout <<"Base Dtor"<<endl; }

};

class Derived: public Base { public:

Derived() { cout <<"Derived Ctor"<<endl; }

~Derived() { cout <<"Derived Dtor"<<endl; }

};

int main() {

Derived d1;

{

Base b1;

}

return 0;

}

D.Base Ctor

Derived Ctor

Base Ctor

Base Dtor

Derived Dtor

Base Dtor

What will be the output of the program?

#include <iostream>

using namespace std;

class F1 {

public:

F1() { cout <<"F1 ctor "; }

~F1() { cout <<"F1 dtor "; }

};

class F2 : public F1 {

public:

F2() { cout <<"F2 ctor "; }

~F2() { cout <<"F2 dtor "; }

};

class F3 : public F1 {

const F2 &f2;

public:

F3() : f2(\*new F2) { cout <<"F3 ctor "; }

~F3() { cout <<"F3 dtor "; }

};

int main() {

F3 f3;

return 0;

}

A.F1 ctor F2 ctor F3 ctor F3 dtor F2 dtor F1 dtor

B.F1 ctor F1 ctor F2 ctor F3 ctor F3 dtor F1 dtor

C.F1 ctor F3 ctor F3 dtor F1 dtor

D.F1 ctor F1 ctor F2 ctor F3 ctor F3 dtor F2 dtor F1 dtor F1 dtor

What will be the output of the program?

#include <iostream>

using namespace std;

class Room {

int number;

public:

Room(int num = 0): number(num) { }

void dimension() { cout <<number <<"Rooms "; }

};

class Building {

public:

Building() : ro(100) { }

void Build() { ro.dimension(); }

private:

Room ro;

};

int main() {

Building B;

B.Build();

return 0;

}

A.0 Rooms

B.100 Rooms

C.Compilation Error: ro is private

D.None of the above

What will be the output of the program?

#include<iostream>

using namespace std;

class Shape {

public:

int x, y;

Shape(int a = 0, int b = 0): x(a), y(b) {}

void draw()

{ cout <<x <<""<<y <<""; }

};

class Rectangle : public Shape {

public:

int w, h;

Rectangle(int a = 5, int b = 6): w(a), h(b), Shape(7, 8) {}

void draw()

{ Shape::draw(); cout <<w <<""<<h ; }

};

int main() {

Rectangle \*r = new Rectangle(1,2);

r->draw();

return 0;

}

A.0 0 1 2

B.7 8 1 2

C.7 8 5 6

D.0 0 5 6

You cannot change the precedence and associativity of an operator by overloading.

A.TRUE

B.FALSE

When deriving a class from with protected inheritance, public members of the base class become\_\_\_\_\_\_\_\_\_\_\_ members of the derived class, and protected members of the

base class become \_\_\_\_\_\_\_\_\_\_\_\_\_ members of the derived class.

A.protected, protected.

B.public, private

C.private, private

D.Private, protected

Id 91

Question When deriving a class with public inheritance, public members of the base class become \_\_\_\_\_\_\_\_\_\_ members of the derived class, and protected members of the base class become \_\_\_\_\_\_\_\_\_\_\_members of the derived class.

A private, private

B public, protected.

C protected, protected.

D private, protected

Answer B

Marks 2

Unit II

Id 92

Question C++ provides for \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, which allows a derived class to inherit from many base classes, even if the base classes are unrelated.

A Multilevel inheritance

B Single level inheritance

C multiple inheritance

D Hierarchical inheritance

Answer C

Marks 1

Unit II

Id 93

Question \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_is a form of software reuse in which new classes absorb the data and behaviors of existing classes and embellish these classes with new capabilities.

A Data hiding

B Inheritance

C Abstraction

D encapsulation

Answer B

Marks 1

Unit II

Id 94

Question We can create the object of abstract class

A True

B False

C

D

Answer B

Marks 1

Unit II

Id 95

Question All virtual functions in an abstract base class must be declared as pure virtual functions.

A TRUE

B FALSE

C

D

Answer B

Marks 1

Unit II

Id 96

Question A class is made abstract by declaring that class virtual.

A TRUE

B FALSE

C

D

Answer B

Marks 1

Unit II

Id 97

Question Polymorphic programming can eliminate the need for switch logic.

A TRUE

B FALSE

C

D

Answer A

Marks 1

Unit II

Id 98

Question Suppose a and b are integer variables and we form the sum a + b. Now suppose c and

D are floating-point variables and we form the sum c + d. The two +operators here are

clearly being used for different purposes. This is an example of \_\_\_\_\_\_\_\_\_\_\_\_

A Operator Overloading

B Inheritance

C Function Overloading

D Constructor

Answer A

Marks 1

Unit II

Id 99

Question The operators that cannot be overloaded is

A \*

B -

C ::

D ()

Answer C

Marks 1

Unit II

Id 100

Question The operators that cannot be overloaded is

A \*

B ?:

C >>

D <<

Answer B

Marks 1

Unit II

Id 101

Question Which of the following operator(s) can not be overloaded

A .\*

B ::

C ?:

D All of the above

Answer D

Marks 1

Unit II

Id 102

Question Which of the following is true about this pointer?

A It is passed as a hidden argument to all function calls

B It is passed as a hidden argument to all non-static function calls

C It is passed as a hidden argument to all static functions

D None of the above

Answer B

Marks 1

Unit II

Id 103

Question Predict the output of following C++ program.

#include<iostream>

using namespace std;

class Test

{

private:

int x;

public:

Test(int x = 0) { this->x = x; }

void change(Test \*t) { this = t; }

void print() { cout <<"x = "<<x <<endl; }

};

int main()

{

Test obj(5);

Test \*ptr = new Test (10);

obj.change(ptr);

obj.print();

return 0;

}

A x = 5

B x = 10

C Compiler Error

D Runtime Error

Answer C

Marks 2

Unit II

Id 104

Question Which of the followings is/are automatically added to every class, if we do not write our own.

A Copy Constructor

B Assignment Operator

C A constructor without any parameter

D All of the above

Answer D

Marks 2

Unit II

Id 105

Question What is the output of following program?

#include<iostream>

using namespace std;

class Point {

Point() { cout <<"Constructor called"; }

};

int main()

{

Point t1;

return 0;

}

A Compiler Error

B Runtime Error

C Constructor called

D Segmentation Fault

Answer A

Marks 1

Unit II

Id 106

Question What will be the output of following program?

#include <iostream>

using namespace std;

class Test

{

public:

Test() { cout <<"Hello from Test() "; }

} a;

int main()

{

cout <<"Main Started ";

return 0;

}

A Main Started

B Main Started Hello from Test()

C Hello from Test() Main Started

D Compiler Error: Global objects are not allowed

Answer C

Marks 2

Unit II

Id 107

Question Which of the following operators are overloaded by default by the compiler?

1) Comparison Operator ( == )

2) Assignment Operator ( = )

A Both 1 and 2

B Only 1

C Only 2

D None of the two

Answer C

Marks 1

Unit II

Id 108

Question A normal C++ operator that acts in a special way on newly defined data types is called \_\_\_\_\_\_\_

A Encapsulated

B Overloaded

C Classified

D Inherited

Answer B

Marks 1

Unit II

Id 109

Question The correct function name for overloading the addition + operator is \_\_

A Operator \_+

B Operator :+

C Operator (+)

D Operator +

Answer D

Marks 1

Unit II

Id 110

Question Which of the following operators cannot be overloaded?

A → operator

B . operator

C [ ] operator

D &operator

Answer B

Marks 1

Unit II

Id 111

Question Which of the following operators cannot be overloaded?

A +

B -

C [ ]

D ::

Answer D

Marks 1

Unit II

Id 112

Question Pick the incorrect statement from the following

A The overloaded operators follow the syntax rules of original operator.

B Only existing operators can be overloaded

C Overloaded operator must have at least one operand of its class type

D Overloaded operators can change the meaning of the original operator

Answer D

Marks 1

Unit II

Id 113

Question For operators to be overloaded as non static member functions:

A Both binary and unary operators take one argument.

B Binary operators can have one argument and unary operators can not have any

C Neither binary nor unary operators can have arguments

D Binary operators can have two arguments and unary operators can have one

Answer B

Marks 1

Unit II

Id 114

Question Which of the following is an operator function?

A Member overloading

B Function overloading

C Operator overloading

D None of these

Answer C

Marks 1

Unit II

Id 115

Question Operator overloading means \_\_\_\_\_\_\_

A Giving new meaning to existing operator without changing its original

Meaning

B Making C++ operators to work with objects

C Making new types of operator

D Both a and b

Answer D

Marks 1

Unit II

Id 116

Question For overloading =+ implicitly \_\_\_\_\_\_\_

A + and = operators need to be overloaded implicitly

B Only + operator need to be overloaded implicitly

C Only = operator need to be overloaded implicitly

D The += operator cannot be overloaded implicitly

Answer D

Marks 1

Unit II

Id 117

Question Overloading a postfix increment operator by means of a member function takes-------

A No argument

B One argument

C Two arguments

D Three Arguments

Answer A

Marks 1

Unit II

Id 118

Question If you overload only prefix operator ++ then the postfix ++ operator is \_\_\_\_\_\_

A Does not work

B Works arbitrarily

C Works naturally

D Works as if prefix ++ operator

Answer D

Marks 1

Unit II

Id 119

Question When compiler decides binding of an overloaded member then it is called\_\_\_\_\_\_\_\_

A Static binding

B Dynamic binding

C Local binding

D None of these

Answer A

Marks 1

Unit II

Id 120

Question One can redefine the working of \_\_\_\_\_\_\_ to work with objects.

A Preprocessor directives

B White space characters

C Standard operators

D None of these

Answer C

Marks 1

Unit II

Id 121

Question Choose the correct option:

I. When you overload <<operator the >>operator automatically gets

overloaded

II. You can overload unary operator to work with binary operator

A Only I is true

B Only II is true

C Both I and II are true

D Neither I nor II are true

Answer D

Marks 1

Unit II

Id 122

Question Choose the correct option

I.If you do not want to make use of operator overloading, you can achieve that effect using user defined function

II. The sizeof operator can be overloaded

A Only I is true

B Only II is true

C Both I and II are true

D Neither I nor II are true

Answer A

Marks 1

Unit II

Id 123

Question The array subscript operator [] when overloaded cannot \_\_\_\_\_\_

A Take user defined objects are operands

B Take float as an operand

C Take multiple values inside (for example: [5,7] )

D None of these

Answer C

Marks 1

Unit II

Id 124

Question The prototype of overloaded cast operator functions do not \_\_\_\_\_\_\_

A specify the type they convert to

B specify the return type

C need to be defined inside the class whose objects are being converted

D none of these

Answer B

Marks 1

Unit II

Id 125

Question Which of the following operators cannot be overloaded ?

A +=

B <<

C ?:

D FUNCTION CALL()

Answer C

Marks 1

Unit II

Id 126

Question Which of the following operators cannot be overloaded ?

A ::

B Sizeof

C Conditional operator ?:

D All of these

Answer D

Marks 1

Unit II

Id 127

Question The overloading the function operator\_\_\_\_\_\_\_\_.

A requires class with overloaded operators

B makes use of parameterized constructor

C allows to create objects that are syntactically like functions

D none of these

Answer A

Marks 1

Unit II

Id 128

Question Choose the incorrect statement from the following.

A Constructors can be overloaded.

B Only existing operators must be overloaded

C the overloaded operators must follow the syntax rules of the original operator

D The overloaded operators must have atleast one operand of its class type

Answer B

Marks 1

Unit II

Id 129

Question Overloading without explicit arguments to an operator function is called\_\_\_\_\_\_.

A unary operator

B binary operator

C nested class

D none of these

Answer A

Marks 1

Unit II

Id 130

Question In binary overloaded function which are overloaded through friend function take\_\_\_\_\_\_\_

A three explicit arguments

B two explicit arguments

C one explicit argument

D no argument

Answer B

Marks 1

Unit II

Id 131

Question In binary overloaded function which are overloaded through member function take\_\_\_\_\_\_\_\_\_\_

A three explicit arguments

B two explicit arguments

C one explicit argument

D no argument

Answer C

Marks 1

Unit II

Id 132

Question The unary operators are overloaded by member function then it takes \_\_\_\_\_\_

A three explicit arguments

B two explicit arguments

C one explicit argument

D no argument

Answer D

Marks 1

Unit II

Id 133

Question Choose the correct choice.

I. All the operators in C++ can be overloaded.

II. We can change the basic meaning of operator while overloading it.

A Only I is true

B Only II is true

C Both I and II are true

D Neither I nor II are true

Answer D

Marks 1

Unit 2

Id 134

Question Which of the following operator can be overloaded through friend function ?

A ::

B +

C =

D ->

Answer B

Marks 1

Unit II

Id 135

Question The name of the operator function that overloads the / symbol is\_\_\_\_\_\_\_\_.

A operator /()

B /op()

C / operator()

D op/()

Answer A

Marks 1

Unit II

Id 136

Question In binary operator overloaded operator function the second operand should be\_\_\_\_\_\_.

A passed by value

B Implicit

C passed by reference

D none of these

Answer C

Marks 1

Unit II

Id 137

Question Function overloading is run time polymorphisms

A True

B False

C

D

Answer B

Marks 1

Unit II

Id 138

Question Following overloaded operator cannot be inherited by derived class\_\_\_\_\_\_\_.

A >

B =

C \*

D /

Answer B

Marks 1

Unit II

Id 139

Question Choose the correct choice.

A The conditional operator can be overloaded

B While overloading using the friend function the binary operator requires one argument

C Operator precedence cannot be changed

D None of these

Answer C

Marks 1

Unit II

Id 140

Question Which of the following operator can be overloaded through friend function ?

A ()

B []

C ->

D \*

Answer D

Marks 1

Unit II

Id 141

Question When we overload we want to\_\_\_\_\_\_.

A compare and copy object

B assign one object to another

C compare two objects

D test for equality

Answer B

Marks 1

Unit II

Id 142

Question Operator overloading is also called one form of polymorphism because\_\_\_\_\_\_\_.

A the overloaded operators have many forms

B the overloaded operators can be declared virtual

C the overloaded function can perform various tasks depending upon the type of object

D None of these

Answer C

Marks 1

Unit II

Id 143

Question Overloading means

A two or more methods in the same class that have same name

B calling the method which has actual parameters

C two or more methods having same name but present in different class

D none of these

Answer C

Marks 1

Unit II

Id 144

Question The inheritance mechanism provides meaning of deriving\_\_\_\_\_\_

A new operator from exciting one

B new function from exciting one

C new class from exciting one

D all of these

Answer C

Marks 1

Unit II

Id 145

Question A class derived from the exciting class is known as\_\_\_\_\_\_

A new class

B Inheritee

C derived class

D none of these

Answer C

Marks 1

Unit II

Id 146

Question The derived class is derived from\_\_\_\_\_\_\_\_\_\_

A derived class

B base class

C both a&b

D none of these

Answer B

Marks 1

Unit 2

Id 147

Question Which of the following can be derived from base class in inheritace ?

A data members

B member function

C both a&b

D none of these

Answer C

Marks 1

Unit II

Id 148

Question The inheritance is described as a \_\_\_\_\_ relationship

A has a

B is a

C association

D none of these

Answer B

Marks 1

Unit II

Id 149

Question Which of the following allows you to create derived class that inherits properties from more than one base class ?

A multilevel inheritance

B multiple inheritance

C single inheritance

D Hybrid inheritance

Answer B

Marks 1

Unit II

Id 150

Question The principle by which the knowledge of general category can be applied to more specific objects is called \_\_\_\_\_

A polymorphism

B overriding

C inheritance

D none of these

Answer A

Marks 1

Unit II

Id 151

Question Parent:child is\_\_\_\_\_\_\_\_

A base:derived

B derived:driven

C child:super

D subclass:superclass

Answer A

Marks 1

Unit II

Id 152

Question What is the syntax of inheritance of a class ?

A Class class \_name

B Class name:access specifier

C Class name:access specifier class name

D none of these

Answer C

Marks 1

Unit II

Id 153

Question If an attribute is private then which methods have access to it?

A Only static methods in the same class

B Only the methods defined in that class

C Only the methods of the same package

D none of these

Answer B

Marks 1

Unit II

Id 154

Question Which of the following advantage cannot be achieved by using multiple inheritance?

A polymorphism

B dynamic binding

C both a&b

D none of these

Answer C

Marks 1

Unit II

Id 155

Question Which of the symbol used to create multiple inheritance ?

A Dot

B Comma

C Hash #

D Dollar

Answer B

Marks 1

Unit II

Id 156

Question Using multiple inheritance

A there can be virtual class

B it can not be include virtual class

C the base classes must have only default constructor

D none of these

Answer A

Marks 1

Unit II

Id 157

Question The\_\_\_\_\_\_member function is declared in base class but redefined in derived class

A class

B overloaded

C operator

D virtual

Answer D

Marks 1

Unit II

Id 158

Question In public inheritance \_\_\_\_\_\_

A All the members of base class are inherited and are made public

B Members of base class that are not private are inherited and retain their access type

C All the members of base class are inherited and retain their access type

D Only public members of base class are inherited and they remain public

Answer C

Marks 1

Unit II

Id 159

Question If class C is derived class of class B and class B is a derived class of A. If we instantiate class B object then the first constructor called belongs to class

A A

B B

C can be A or B

D one cannot achieve such inheritance

Answer A

Marks 1

Unit II

Id 160

Question When the object of derived class expire, first the \_\_\_\_\_\_\_ is invoked followed by the \_\_\_\_\_\_\_\_.

A derived class constructor, base class destructor

B derived class destructor , base class destructor

C base class destructor , derived class destructor

D none of these

Answer B

Marks 1

Unit II

Id 161

Question If class A inherits from class B then B is called \_\_\_\_\_\_\_\_ ans A is called \_\_\_\_\_\_\_\_ of B.

A superclass and subclass

B subclass and superclass

C subclass and child class

D superclass and parent class

Answer A

Marks 1

Unit II

Id 162

Question What does derived class does not inherit from the base class \_\_\_\_\_\_\_.

A constructor and destructor

B operator=() members

C friends

D all of these

Answer D

Marks 1

Unit II

Id 163

Question Which constructor will initialise the base class data member ?

A Base class

B Derived class

C Derived derived class

D None of these

Answer A

Marks 1

Unit II

Id 164

Question If class A is a friend class of class B, if class B is friend class of class C then\_\_\_\_\_\_\_

A class C is friend class of

B class A is friend class of

C class A and class C do not have any friendship relation.

D none of these

Answer C

Marks 1

Unit II

Id 165

Question \_\_\_\_\_\_\_\_\_ class is tightly coupled with other class.

A friend

B virtual

C abstract

D none of these

Answer A

Marks 1

Unit II

Id 166

Question The keyword friend is used in \_\_\_\_\_\_\_\_\_\_\_.

A the class allowing access to another class

B the private section of a class

C the public section of a class

D all of these

Answer D

Marks 1

Unit II

Id 167

Question Class Test:public A, public B is an example of multiple inheritance.

A False

B True

C

D

Answer B

Marks 1

Unit II

Id 168

Question Which of the following interface determines how your program will be used by other program?

A Public

B Private

C Protected

D None of these

Answer A

Marks 1

Unit II

Id 169

Question When base class pointer points to derived class object\_\_\_\_\_\_\_\_\_\_\_

A it can access only base class members

B it can access only derived class members

C both base class &derived class members

D None of these

Answer A

Marks 1

Unit II

Id 170

Question The base class will offer\_\_\_\_\_

A more specific object than the derived class

B more generalized version of its derived class

C empty templates of its derived class

D none of these

Answer B

Marks 1

Unit II

Id 171

Question In my program I have overloaded TEST::operator+ and TEST::operator= What is the effect on TEST::operator+= ?

A The TEST::operator+= will be automatically overloaded .first TEST::operator+ will get overloaded and then TEST::operator=

B The TEST::operator+= will be automatically overloaded .first TEST::operator= will get overloaded and then TEST::operator+

C TEST::operator+= will made invalid

D There will be no effect because all three are independent

Answer D

Marks 1

Unit II

Id 172

Question What will happen on execution of the following code ?

Class base

{

};class derived: protected base

{

};

A It will not compile as the class body of the base class is not defined

B It will not compile as the class body of the derived class is not defined

C It will compile successfully

D The compilation of above code is dependent upon the type of data provided to it

Answer C

Marks 1

Unit II

Id 173

Question The base class will offer\_\_\_\_\_

A more specific object than the derived class

B more generalized version of its derived class

C empty templates of its derived class

D none of these

Answer B

Marks 1

Unit II

Id 174

Question The hybrid inheritance is \_\_\_\_\_

A multiple inheritance

B multilevel inheritance

C multipath inheritance

D both a &c

Answer D

Marks 1

Unit II

Id 175

Question Ho

w many types of inheritance are there

A 1

B 2

C 4

D 5

Answer D

Marks 1

Unit II

Id 176

Question Choose the correct option

A a) a constructor can not be called explicitly

B b) a destructor is not inherited

C c) constructor can not be inherited

D d) All of these

Answer D

Marks 1

Unit II

Id 177

Question Suppose class derived is derived from a class Base. Both the classes contain the

Function name display() that take no argument. What will be the statement in the class derived which will called the display function of base class

A Display()

B Base:display()

C Base ::display()

D Can make such a cell

Answer C

Marks 1

Unit II

Id 178

Question Suppose class derived is derived from a class Base privately. The object of class Derived is located in main() can access\_\_\_\_\_\_\_.

A public members of base

B private member of base

C protected members of base

D public members of derived

Answer D

Marks 1

Unit II

Id 179

Question Multiple inheritance causes for a derived class to have \_\_\_members.

A ambiguous

B public

C private

D protected

Answer A

Marks 1

Unit II

Id 180

Question What will be the first line of specifier for the class tier, wheel &rubber. Make use of public rubber

A Class Tier:public wheel, public rubber

B Class wheel:public tier, public rubber

C Class rubber:public tier, public wheel

D none of these

Answer A

Marks 1

Unit II

Id 181

Question Which is the correct class defination for class C ,Which inherits from A &B classes

A Class C:A,B

B Class C::A,B

C Class C:public A,public B

D Class C:: public A,Public B

Answer C

Marks 1

Unit II

Id 182

Question The ability of function or operator to act in different ways on different data type is called\_\_\_\_\_\_\_\_\_\_\_

A inheritance

B polymorphism

C encapsulation

D none of these

Answer B

Marks 1

Unit II

Id 183

Question \_\_\_\_\_class that declares or inherits a virtual function.

A Encapsulation data

B Inherited class

C Polymorphic class

D none of these

Answer C

Marks 1

Unit II

Id 184

Question Choose the correct option.

A A base class may have more then one derived class

B Derived class may have more than one derived class

C Both a &b

D Neither a nor b

Answer C

Marks 1

Unit II

Id 185

Question reusability is provided by which feature of c++

A polymorphisms

B abstraction

C derivation

D none of the above

Answer C

Marks 1

Unit II

Id 186

Question What types of derivations are supported by c++?

A single

B multiple

C multilevel

D all of the above

Answer D

Marks 1

Unit II