FINALTERM EXAMINATION

Fall 2009

CS201- Introduction to Programming

Time: 120 min
Marks: 75
If we write a statement like **s2 = s1;** ___ will be the calling object and ____ will be passed to the = operator as an argument.

- ► s1, s1
- ► s1, s2
- ► **s2, s1**
- ► s2, s2

If we write a statement like *s2 = s1*; *s2* will be the calling object and *s1* will be passed to the = operator as an argument. P# 397
cout << setfill('0') << setw(7) << 128 ;

- ► **0000128**
- ► 0128128
- ► 1280000
- ► 0012800

default alignment is from left due to this it first prints 4 Zeros(setw=7, digit=3 i.e 1-2-8,) 7-3=4 Zeros vuzs

The stream insertion and extraction operators are not already overloaded for _____

- ► Built-in data types
- ► **User-defined data types**
- ► Both built-in and user-defined types
- ► None of the given options

Constructors can not be overloaded like ordinary functions.

- ► True
- ► **False**

The constructors can be overloaded. We can write as many constructors as we

require. At one time, the compiler will call the correct version of the constructor".P# 323 these solutions are meant to host at vuzs site only

Overloaded new operator function takes parameter of type *size_t* and returns

- ► void (nothing)
- ► void pointer
- ► object pointer
- ► int pointer

Also note that the *new* operator returns a *void* pointer. Any *new* operator we write must have this parameter and return type.

Which of the following is the correct way to declare a variable x of integer type?

- ► x int ;
- ► integer x ;
- ► **int x;**
- ► x integer

Reserve words cannot be used as a variable name.

- ► **True**
- ► False

There are few data types in C language. These data types are reserved words of C language. The reserve words can not be used as a variable manes. P# 17

A template function must have at least ---------- generic data type

- ► Zero
- ► **One**
- ► Two
- ► Three

The function arguments must contain at least one generic data type. P# 499

Template functions can also be overloaded

- ► True
- ► **False**

We can write overloaded template functions as long as there is use of different number or type of arguments.. P # 503

We can not make a member function of a class as template function.

- ► True
- ► **False not sure**

When break statement is encountered in switch statement, it

- ► Stops the entire program
- ► **Stops the execution of current statement**
- ► Exits from switch statement
- ► None of the given options

We can also define a variable of user define data type (object) as static.

- ► **True**
- ► False

The declarator of Plus (+) member operator function is

- ► **Class-Name operator + (Class-Name rhs)**
- ► operator Class-Name + ( )
- ► operator Class-Name + (rhs)
- ► Class-Name operator + ( )

Let suppose

```
int a, b, c, d, e;
a = b = c = d = e = 42;
```

This can be interpreted by the complier as:

► a = (b = (c = (d = (e = 42))));
► (a = b = (c = (d = (e = 42))));
► **a = b = (c = (d = (e = 42)));**
► (a = b) = (c = d) = (e = 42);

a = (b = (c = (d = (e = 42))));

What will be the range of numbers generated by function rand () % 9?

► 0 to 9
► 1 to 9
► **0 to 8**
► 1 to 8

When 6 divides any number, the remainder will always be less than 6. Ther result will be between therefore we will add 1. 1 + rand ( ) % 6;

Which of the following is the correct function call having array named *student* of 10 elements as a parameter.

► **addRecord(student[]) ;**
► addRecord(student) ;
► addRecord(student[10]) **;**
► addRecord(*student) ;

when we pass array we don't give limit of array

Example:

Pass array to function

http://vua

```
#include<stdio.h>
   #include<conio.h>
   void read(int *,int);
   void dis(int *,int);

   void main()
   {
    int a[5],b[5],c[5],i;

    printf("Enter the elements of first list \n");
    read(a,5);
    printf("The elements of first list are \n");
    dis(a,5);
   }

   void read(int c[],int i)
   {
    int j;
    for(j=0;j<i;j++)
      scanf("%d",&c[j]);
    fflush(stdin);
   }

   void dis(int d[],int i)
   {
    int j;
    for(j=0;j<i;j++)
    printf("%d ",d[j]);
    printf("\n");
   }
```

Declaring structures does not mean that memory is allocated.
Example:

- **► True**
- ► False

structures do not occupy any memory until it is associated with the structure variable

Identifier is a name that can be given to variables, labels and functions.

- **► True**
- ► False

An 'Identifier' means any name that the user creates in his/her program. These names can be of variables, functions and labels

If a class A declares itself a friend of class B and a class B declares itself a friend of class C then

► Class A is also a friend of class C.
► Class B is also a friend of class A.
► **Class A is also a friend of class C if A declares C as its friend.**
► Class A is also a friend of class C if C declares A as its friend.

If we want a two-way relationship, OtherClass will have to declare ClassOne as a friend class, resulting in a complete two-way relationship

Which of the following statement is best regarding declaration of friend function?

► Friend function must be declared after public keyword.
► Friend function must be declared after private keyword.
► Friend function must be declared at the top within class definition.
► **It can be declared anywhere in class as these are not affected by the public and private keywords**.

Friend is a very strong statement. It is too strong to be affected by public or private we can put it anywhere in the class

A pointer is a special type of variable that contain _____

► **Memory Address**
► Data values
► Both Values and Memory
► None of given of options

Pointer is a special type of variable that contains a memory address.

When memory for a program is allocated at run time then it is called _____

► static memory allocation
► **dynamic memory allocation**
► stack memory allocation
► virtual memory allocation

When we create an object of the class at run time, it will allocate memory according to our requirement. So there is no waste of memory and the situations in which we want to store large data in small memory or vice versa are prevented. So we do dynamic memory allocation inside these classes.

What purpose do classes serve?

- ► Data encapsulation
- ► Providing a convenient way of modeling real-world objects
- ► Simplifying code reuse
- ► **All of the given options**

Which of the following function cannot be overloaded?

- ► Member functions
- ► Utility functions
- ► Constructor
- ► **Destructor**

The destructors can be summarized as The destructors cannot be overloaded. The destructors take no arguments. The destructors don't return a value

The following prototype of unary operator function indicates that it is _____ .

Date operator++(int)

- ► **Member functions of post increment operator**
- ► Member functions of pre increment operator
- ► Non-member functions of post increment operator
- ► Non-member functions of pre increment operator

Overloading Unary Operators

```
// Preincrement operator overloaded as a member function.
Date Date::operator++()
{
   helpIncrement();
   return *this;  // value return; not a reference return
}

// Postincrement operator overloaded as a member function.
// Note that the dummy integer parameter does not have a
// parameter name.
Date Date::operator++(int)
{
   Date temp = *this;
   helpIncrement();

   // return non-incremented, saved, temporary object
   return temp;   // value return; not a reference return
} // This paper was solved by vuzs Team and meant for hosting
at vuzs otherwise its stolen contents
```

Static variable which is defined in a function is initialized _____.

- ► **Only once during its life time**
- ► Every time the function call
- ► Compile time of the program
- ► None of the above

Once the static variables are created, they exist for the life of the program. They do not die.

In the member initialize list, the data members are initialized,

- ► **From left to right**
- ► From right to left
- ► In the order in which they are defined within class
- ► None of the given options

If we do not indent the code properly it will _____

- ► Be a syntax error
- ► Be a logical error
- ► **Not be an error at all**
- ► None of the given options

we Indent the code for better readability and understanding

Truth tables are used for analyzing _____.

- ► **logical expressions**
- ► arithmetic expressions
- ► both logical and arithmetic expressions
- ► none of the given options.

The truth tables are very important. These are still a tool available for analyzing logical expressions.

Static memory allocation is also known as _____

- ► Dynamic allocation
- ► **Compile time allocation**
- ► Run time allocation
- ► None of the given options

This type of memory static allocation. It is also known as compile time allocation.

(Marks: 1)

What does *getline()* member function of cin stream do?

Another member function of *cin* is *getline()*. It reads a complete buffer i.e. the number of character specified up to a delimiter we specify. We can write something like:

cin.getline(char *buffer, int buff_size, char delimiter = '\n')

(Marks: 1)

When memory is allocated dynamically using new operator within the constructor of class then what is an appropriate place to de-allocate the memory?

Whenever we allocate memory with the *new* operator, it is our responsibility to de-allocate this memory after the termination of the program. To do this de-allocation, we have an operator *delete*. To de-allocate the memory, allocated with *p = new int ;* we will write delete

(p) ;

It will not delete the *p* rather, it will send the memory gotten and pointed by *p* back to the free store.

(Marks: 2)

What will be the output of following code, if user input a number **123**?

```
int input ;
cin >> oct >> input;
cout << hex << input ;
```

(Marks: 2

What is memory leak?

suppose, the heap size is decreased as we had allocated memory from it despite the fact that it was never utilized. If this step of allocating memory and then destroy the pointer to this memory carries on then the size of the heap will going on to decrease. It may become of zero size. When there is no memory on heap, the computer will stop running and there may be a system crash. This situation is called a memory leak

(Marks: 3)

When we call calloc function to allocate memory and its return a NULL pointer what does it mean?

Calloc function takes two arguments. The first argument is the required space in terms of numbers while the second one is the size of the space

Now we have to see what happens when either we ask for too much memory at a time of non-availability of enough memory on the heap or we ask for memory that is available on the heap , but not available as a single chunk?. In this case, the call to calloc will fail. When a call to memory allocation functions fails, it returns a NULL pointer.

(Marks: 3)

Read the given code and explain code functionality.

```
Matrix :: Matrix ( const Matrix & m )
{
   numRows = m.numRows ;
   numCols = m.numCols ;
   elements = new ( double * ) [ numRows ] ;
   for ( int  i = 0 ; i < numRows ; i ++ )
{
      elements [ i ] = new double [ numCols ] ;
      for ( int j = 0 ; j < numCols ; j ++ )
            elements [ i ] [ j ] = m.elements [ i ] [ j ] ;
   }
}
```

(Marks: 3)

What is the keyword 'this' and what are the uses of 'this' pointer?

Whenever an object calls a member function, the function implicitly gets a pointer from the calling object. That pointer is known as *this* pointer. '*this*' is a key word. We

cannot use it as a variable name. '*this*' pointer is present in the function, referring to the calling object.
this pointer points to the current object.
(Marks: 5)

What do you mean by garbage collection and how it works in JAVA and C++ ?

JAVA gives the concept of garbage collection with the use of references. Due to this garbage collection, we are free from the headache of de- allocating the memory. We allocate and use the memory. When it is no longer in use, JAVA automatically deletes (frees) it through garbage collection But in C and C++ languages, we have to take care of de-allocating the memory. In classes where we use dynamic memory, we have to provide destructors to free this memory. The languages keep evolving, new constructs will keep evolving in existing or new languages.
(Marks: 5)

Explain the concept of separation of interface from the implementation in the context of classes, using a real world example.
(Marks: 10)

Write a simple program using the **get()** member function of **cin** object reading a text of **30** characters from the keyboard, store them in an array and then using **put()** member function of **cout** object to display them on the screen.
(Marks: 10)

Overload the Binary Assignment (=) Operator.
Write a program which has a class **List**, This class should have Two data members, an array of integers **list[]** and an integer variable **length** (i.e. number of elements in the list).The class should further contain a default constructor, a **Print()** function which display the list and a Function **insert()** which insert an element in the list and Assignment (=) Operator function, which contain code for the assignment of one object to other. .
In main function define two objects **list1** and **list2** and use the statement **list2 = list1;** and use (call) **print** function with both objects