Turgut Tezir
C++ Programming
Final Exam

1. Why should I switch from C to C++?
   - C++ is Object oriented that gives lots of advantage about writing a program. (C programmers concentrated on writing functions. Group of actions that perform some task are formed into functions, and functions are group to forms program. C++ programmers concentrate on creating their own user-difined types called classes. Classes also referred to as programmer define types. Each class contains data as well as the set of functions that manipulate data.) C++ includes everything that is part of C, but the most important is that C++ supports Object Oriented Programming
2. What does OOP stand for? What is the term for non-OOP languages?
   - OOP stand for Object Oriented Programming (Program written that define all objects to include their attributes and behaviors). Non OOP are Structured Programming (Computer System is viewed as a collection of computer program) and Information System Management.

3. Why is dynamic memory allocation becoming even more important as time goes by?
   - Because dynamic memory allocation- has the ability of a program to obtain more memory space at execution time to hold new nodes and to release space no longer needed. The limit for the dynamic memory allocation can be as large as the amount of variable physical memory in the computer or the amount of available virtual memory in a virtual memory system.
   - Memory allocation provides a way to dynamically create buffers and arrays. Dynamic means that the space is allocated in memory as the program is executing

4. What are the C++ equivalents of malloc() and free()?
   - C++ programs can contain storage created by malloc and deleted by free, and objects created by **new** and deleted by **delete**. Is the best use only **new** and **delete.**

5. What is wrong with the following code's outline?
   - //First fart of program here
     int * i=new int; **int *I;**
     //Middle of program **I = new int;**
     free(i); **free (I);**
     //Last part of program

6. What is a heap?
   - Heap: Unused memory in the computer when a program is executed. This unused memory is called the heap. (During program execution dynamically allocated memory comes form a pool of memory known as the heap or free store. It allocated using the C++ operator "new" and freed using the operator "delete".)

7. Write a program with two functions that have the same name. The first function is to return the average of an integer array passed to it, and second function is to return the average of a floating-point array?

```
//Turgut Tezir
//Final # 7
#include<iostream.h>
void calc(int iarray[],int);
void calc(float farray[],int);
void main(void)
{
	int const size =10;
	int iarray[size];
	float farray[size];
	float average =0;
		cout<<"Enter10 nums.integer array:\n";
		for(int i = 0;i < size; i++)
				iarray[i]= i;
			cout<<"\nAverage:\n";
			calc(iarray,size);
			cout << "\nEnter 10 nums.floating point array:\n";
			for(i = 0; i < size; i++)
				farray[i]=1.1;

			cout<<"Average:\n";
			calc(farray,size);
}
void calc(int iarray[],int size)
{
	int average = 0;
	for(int i=0; i < size; i++) {average = average + iarray[i];}
	average = average /size;
	cout << average;
}
void calc(float farray[],int size)
{
	float average = 0;
	for(int i=0; i< size; i++){average += farray[i];}
	average = average / size;
	cout << average;
}
```

8. What is the advantage of overloaded functions?

- Function overloading provides a way to have multiple functions with the same name. The same function name can be used for multiple functions provided each

differs in the number or type of its parameters. For each call of the function, the compiler compares the number and type of the arguments in the call against the parameter lists of each version of the function

9. What is the difference between overloaded functions and default argument list?
   - Overload function two or more functions may have the same name as long as the parameters list are different. Default Arguments are passed automatically to parameter if no argument is provided in the function call.

10. How does C++ differentiate one overloaded function from another?
   - Parameter list must be different.

11. If you wanted to overload the operator &&, ||, and *=, what is the name of the functions you would have to write?

   - void operator =&&( )
     void operator = || ( )
     void operator = *=( )

     (Name of function)

12. What is an abstract data type?
   - ADT are data type created by the programmer. ADT has own range of data and their own set of operations that maybe performed on them. -- It is a model, a structure with a number of operations, which affect this model

13. What is an object?
   - An object is an entity that is able to save a state (information) and which offers a number of operations (behaviors) to either examine or affect this state. Objects correspond to real-life entity such as car, phone, book and etc. The behavior and the information are encapsulated in the object.

14. What is the difference between struct and classes?
   - Classes enable the programmer to model objects with attributes and behaviors. Class types can be defined in C++ using the keywords **class** and **struct** , but the key word **class** is normally used for this purpose.We use structs to group variables into a single record. They are similar but; member of a class are private by default and members of a structure are public by default.

15. Can I keep using the familiar struct keyword because I am most used to it? (I'll keep everything private that should be)
   - If you use structure you have to use **struct**

16. Can I define an abstract data type with a class?
   - Yes, objects are created from programmer-defined abstract data type.

17. Why are objects said to be active rather than passive?
    - Because C++ programmers determine the set of classes that will be used to create the objects that will work together to implement the system. Object Oriented Programming is centered around object.

18. Why do I need constructor and destructor? What is default constructor?
    - We need constructor to initialize data members and allocate member data on the heap when allocation is required. Destructors deallocate heap memory. Destructor also can be use to close files, windows and allocate heap memory. Constructors are called automatically. Destructors are called when variable go out of scope. Default constructor is when a constructor does not have to accept arguments

19. What exactly is a *this pointer?
    - Pointers are variables that contain as their values address of other variable. -- Pointers enable programs to simulate call-by-reference and to create and manipulate dynamic data structures (ex: Data structure that can grow and shrink), such as linked lists, queues, stacks and tress.

20. Describe two ways to call member functions?
    - Message correspond to a member-function call sent from one object to another or send from a function to an object.

21. What is the maximum number of destructors a class can contain?
    - A class can contain at most one destructor function.

22. When are your class constructors and destructors called?
    - Constructors are called automatically when you define an object. Destructors are called when variable go out of scope.

23. Write a beginning of grading program for a teacher. Create a class named LetterGrades that contains a single character data member named grade. Write two public member functions, one that passes a letter grade to the class variable from main() and one that prints the letter grade. Make sure the first function contains a default argument of A so that he program doesn't have to pass an A, just other grades. Define several variables (representing different student scores), and initialize and print them.
    - Add a private member function to the LetterGrades class which ensures that the letter grade falls in the range from A to F (convert lowercase letters a through f to uppercase if needed). Call the member function from the public initialization function before initializing the grade with a value.

```
//Turgut Tezir
//Final # 23
#include <iostream.h>
#include <string.h>
#include <ctype.h>
```

```
class LetterGrades
{
	private:
		char grade;
		char TestGrade(const char);
	public:
		LetterGrades();
		void SetGrade(const char g = 'A');
		void PrintGrade();
};

LetterGrades::LetterGrades(){}

void LetterGrades::SetGrade(const char g)
{
		grade = TestGrade(g);
}

void LetterGrades::PrintGrade()
{
	if (grade == 'N')
		cout<<"not valid";
	else
		cout<<grade;
}

char LetterGrades::TestGrade(const char g)
{
	char testg;
	testg = toupper(g);
	if(testg >= 'A' && testg <= 'F' && testg != 'E')
		return testg;
	else
		return 'N';
}

void main()
{
 char g1 = 'G';
 LetterGrades invalid, defaultgrade;
 defaultgrade.SetGrade();
 cout<<"Default grade is ";
 defaultgrade.PrintGrade();
 invalid.SetGrade(g1);
 cout<<"\nInvalid grade is ";
```

```
  invalid.PrintGrade();
}
```

True-False

24. The following statement both allocates and initializes an integer on the heap to zero:

- T int *myval= new int(0);

25. The following statement both allocates and initializes an integer array on the heap to zero:

- T int *myarray = new int [40] (0);

26. A class can contain more than one constructor function.

- T

27 Pg 450, #6.11,6.12, 6.13

```
#include<iostream.h>
#include<iomanip.h>
class Rectangle
{
        public:
        Rectangle(double=1.0,double=1.0);
        double Perimeter (void);
        double Area (void);
        void Setwidth(double w);
        void Setlength(double l);
        double getwidth(void);
        double getlength(void);
        private:
        double length;
        double width;
        };
        Rectangle::Rectangle(double w, double l)
                {
                        Setwidth (w);
                        Setlength (l);
                }
        double Rectangle::Perimeter (void)
                {
                        return 2*(width+length);
                }
        double Rectangle::Area (void)
                {
                return width*length;
                }
        void Rectangle::Setwidth (double w)
```

```
		{
		width=w>0.0&&w<20.00?w:1.0;
		}
	void Rectangle::Setlength (double l)
		{
		length=l>0.0&&l<20.00?l:1.0;
		}
	double Rectangle::getwidth (void)
		{
		return width;
		}
	double Rectangle::getlength (void)
		{
		return length;
		}

void main (void)
{
	Rectangle a, b(4.0, 5.0), c(67.0, 888.0);
	cout<<setiosflags(ios::showpoint|ios::fixed)
		<<setprecision(1);
	cout<<"a:Length = "<<a.getlength()
		<<";Width = "<<a.getwidth()
		<<";perimeter = "<<a.Perimeter()
		<<";Area = "<<a.Area()<<'\n';
	cout<<"b:length = "<<b.getlength()
		<<";width = "<<b.getwidth()
		<<";perimeter = "<< b.Perimeter()
		<<";Area = "<<b.Area()<<'\n';
	cout<<"C:length= "<<c.getlength()
		<<";width = "<<c.getwidth()
		<<";perimeter = "<<c.Perimeter()
		<<";Area = "<<c.Area();
}
```