OOP344, Test 2 Version 1

Name:

Last Name:

Seneca Email id:

Section:

Determine the exact output of the following program: [10 marks]

#include <iostream>

#include <cstdarg>

#include <fstream>

using namespace std;

// if this code is saved in a file called w1.cpp and the executable is exectued as:

// $ w1 Hee Haa Hoo

void w5(void){

unsigned int tst = 34569;

unsigned int x = ~tst + 1;

cout<<"W5: "<<x+tst<<endl;

}

void w4(const char\* fn){

char ch;

ifstream w(fn,ios::binary);

w.seekg(11);

w.read(&ch,1);

cout<<"W3: "<<ch;

w.seekg((ios::off\_type)-1, ios::cur);

w.read(&ch,1);

cout<<ch;

w.seekg((ios::off\_type)1, ios::beg);

w.read(&ch,1);

cout<<ch;

w.seekg((ios::off\_type)-1, ios::end);

w.read(&ch,1);

cout<<ch<<endl;

}

class A{

private:

int \_data;

public:

A(int data = -1):\_data(data){

}

void print(){

cout<<" A: "<<\_data;

}

};

class B:public A{

public:

B(int data):A(data\*10){}

void print(){

A::print();

}

};

class C:virtual public A{

public:

C(int data = 10):A(data/2){}

void print(){

A::print();

}

};

class D:public B, public C{

public:

D(int data):B(data/2){}

void print(){

B::print();

C::print();

}

};

void w3(int d){

cout<<"W3:";

D(d).print();

cout<<endl;

}

void w2(int n, ...){

int s = 0;

va\_list varg;

va\_start(varg, n);

for(int i =0;i<n;i++){

s += (i%2 == 0) ? va\_arg(varg, int): -va\_arg(varg, int);

}

va\_end(varg);

cout<<"W2: "<<s<<endl;

}

int main(int argc, char\* argv[]){

cout<<argc<<": "<<argv[3]<<endl;

w2(3,40,50,60);

w3(30);

w4("w1.cpp");

w5();

return 0;

}

Programming:

2: Write a program that receives an integer and prints it bit pattern. If the program is compiled under the name “int2bit”, then execution should be as follows

$ int2bit 42 <Enter>  
00000000000000000000000000101010  
$\_

3: Write a template for a standard Queue with following methods

Queue::add(T) ; Adds data to the tail of Queue  
T Queue::remove(); Removes data from the head of Queue and returns it  
bool Queue::isEmpty(); Returns true if the Queue is empty  
(make sure you also do the constructor and destructor of the queue.

4: Using Multiple inheritance Create a “Stateful” Queue out of the Queue in question 3 and fstream class.

A Stateful Queue saves all its “un-removed” data (in its nodes) in a file before destruction, so it can load them back if the Queue is instantiated again.

To implete this Queue (lets call it SQueue) inherit it from Queue<T> and fstream.   
 SQueue<T> works exactly like Queue<T>, so there is no need to modify add(), remove() or isEmpty().

Constructor:  
The Constructor of SQueue<T>, however, receives a file name (const char\*) and passes it to its parent (fstream) along with the necessary flags to open the file for a “binary read” and also keeps the file name in an attribute (lets call it “\_filename”) to be used later in the destructor.   
Then, if the parent (fstream) is in a good state and did not “fail()” (i.e. the file is opened), it reads the data (all the “T”s in the file) from the file and adds them to the queue and then closes the file. Otherwise (if the file was not opened) it does nothing but clearing the state of fstream (i.e. “clear()”).

Destructor:  
The Destructor of SQueue<T> , opens the same file that was opened in the Constructor (“\_filename”) for a “binary write” and then writes all the remaining data of SQueue into the file. (by removing them one by one, to the end and writing them into the file).