CS20A Midterm

NASHIR JANMOHAMED

TOTAL POINTS

54.5 / 59

QUESTION 1

1 1 / 1

✓ **- 0 pts Correct**

- 1 pts False

QUESTION 2

2 1 / 1

✓ **- 0 pts Correct**

- 1 pts True

QUESTION 3

3 1 / 1

✓ **- 0 pts Correct**

- 1 pts False

QUESTION 4

4 1 / 1

✓ **- 0 pts Correct**

- 1 pts True

QUESTION 5

5 1 / 1

✓ **- 0 pts Correct**

- 1 pts False

QUESTION 6

6 0 / 1

- 0 pts Correct

✓ **- 1 pts True**

QUESTION 7

7 1 / 1

✓ **- 0 pts Correct**

- 1 pts False

QUESTION 8

8 1 / 1

✓ **- 0 pts Correct**

- 1 pts False

QUESTION 9

9 1 / 1

✓ **- 0 pts Correct**

- 1 pts False

QUESTION 10

10 1 / 1

✓ **- 0 pts Correct**

- 1 pts False

QUESTION 11

11 0 / 1

- 0 pts Correct

✓ **- 1 pts False**

QUESTION 12

12 1 / 1

✓ **- 0 pts Correct**

- 1 pts True

QUESTION 13

13 1 / 1

✓ **- 0 pts Correct**

- 1 pts False

QUESTION 14

14 1 / 1

✓ **- 0 pts Correct**

- 1 pts False

QUESTION 15

15 1 / 1

✓ - 0 pts Correct

- 1 pts False

QUESTION 16

16 0 / 1

- 0 pts Correct

✓ - 1 pts False

QUESTION 17

17 1 / 1

✓ - 0 pts Correct

- 1 pts True

QUESTION 18

18 1 / 1

✓ - 0 pts Correct

- 1 pts True

QUESTION 19

19 1 / 1

✓ - 0 pts Correct

- 1 pts False

QUESTION 20

20 1 / 1

✓ - 0 pts Correct

- 1 pts False

QUESTION 21

21 1 / 1

✓ - 0 pts Correct

- 1 pts d.

QUESTION 22

22 1 / 1

✓ - 0 pts Correct

- 1 pts d

QUESTION 23

23 1 / 1

✓ - 0 pts Correct

- 1 pts b

QUESTION 24

24 1 / 1

✓ - 0 pts Correct

- 1 pts b

QUESTION 25

25 1 / 1

✓ - 0 pts Correct

- 1 pts c

QUESTION 26

26 1 / 1

✓ - 0 pts Correct

- 1 pts a

QUESTION 27

27 1 / 1

✓ - 0 pts Correct

- 1 pts e

QUESTION 28

28 1 / 1

✓ - 0 pts Correct

- 1 pts a

QUESTION 29

29 1 / 1

✓ - 0 pts Correct

- 1 pts d

QUESTION 30

30 30 a 3 / 3

- 0.5 pts arr[0]:4

- 0.5 pts arr[1]:79

- 0.5 pts arr[2]:5

- 0.5 pts arr[3]:9

- 0.5 pts arr[4]:-1

- 0.5 pts arr[5]:19

✓ - 0 pts Correct

- 3 pts Incorrect

- 0.25 pts missing minus sign

QUESTION 31

31 30 b **2 / 2**

✓ **- 0 pts Correct**

- 2 pts Incorrect

- 1 pts Swap1

- 1 pts Dereference Pointers

- 0.5 pts Explanation unclear

QUESTION 32

32 30 c **2 / 2**

✓ **- 0 pts Correct**

- 2 pts Incorrect

- 0 pts Wrong swap

- 1 pts Syntax

- 1 pts Syntax

- 1 pts Logic Error

- 1 pts use swap as identifier

QUESTION 33

33 31 a **2 / 2**

✓ **- 0 pts Correct**

- 0.5 pts newline

- 1 pts 1 Partial

- 1.5 pts partial

- 2 pts Incorrect

QUESTION 34

34 31 b **2 / 3**

- 0 pts Correct

- 1 pts D'tor

- 1 pts oper'=

✓ **- 1 pts Copy C'tor**

- 3 pts Incorrect

QUESTION 35

35 31 c **2 / 2**

✓ **- 0 pts Correct**

- 0.5 pts Syntax

- 0.5 pts Logic

- 0 pts for() delete p{}

- 1 pts delete [] p

- 2 pts incorrect

QUESTION 36

36 31 d **2 / 2**

✓ **- 0 pts Correct**

- 1 pts a = other.a;b=other.b;

- 1 pts p=new int*[a]; for() p[]

- 0.5 pts Syntax

- 0.5 pts Logic

- 2 pts incorrect

QUESTION 37

37 31 e **4 / 4**

✓ **- 0 pts Correct**

- 0.5 pts Check for self assignment

- 1 pts a=other.a; b=other.b;

- 1 pts for() delete p[]; delete []p

- 1 pts p=new int[] for() p[] = new

- 0.5 pts return *this;

- 0.5 pts Syntax

- 0.5 pts Logic

- 4 pts Incorrect

QUESTION 38

38 32 a **2 / 2**

✓ **- 0 pts Correct**

- 1 pts virtual destructor

- 1 pts virtual print

- 2 pts Incorrect

QUESTION 39

39 32 b **3 / 3**

✓ **- 0 pts Correct**

- 1 pts Reuse

- 1 pts Extend

- 1 pts Specialize

- 1 pts Specific examples

- 0.5 pts Logic

- 3 pts Incorrect

QUESTION 40

40 32 c **1.5 / 2**

- 0 pts Correct

- 0.5 pts Check for self assignment

- **0.5 pts** Call base assignment
- **0.5 pts** delete q; new q...
- **0.5 pts** return *this;
- **0.5 pts** Syntax

✓ **- 0.5 pts** **Logic**

- **2 pts** Incorrect

QUESTION 41

41 33 **3 / 3**

✓ **- 0 pts** **Correct**

- **1 pts** P P P P D F C
- **0.5 pts** Wrong construction order/ or number of calls
- **0.5 pts** b r
- **0.5 pts** B r
- **1 pts** ~C ~F ~D ~P ~P ~P ~P
- **0.5 pts** Wrong destruction order/ or number of calls
- **0.5 pts** Overall Order
- **0.5 pts** Pluto does not construct anything.
- **3 pts** Incorrect

Name: Nashir Jannohamed ID: 1325605

Santa Monica College

CS20A Data Structures with C++

Midterm Exam

Spring 2018

Closed Book, Closed Notes, No Electronic Devices.

Please write your name on every page. Read each problem carefully.
You may and are encourage to use scratch paper to organize your thoughts.
However, your answer must be clearly indicated on the exam itself.
The scratch paper will not be considered in grading.

Good Luck!

Name: Nashir Janmohamed

True/False: Write out either True or False.

1. F The implicit default constructor will initialize member variables to default values.
2. T A local variable can be used only within the statement block in which it is declared.
3. F Not defining a destructor for your class will result in a compiler error.
4. T If unstated, the default access modifier of a class in C++ is private.
5. F Declaring a pointer to an object will call the constructor for that object.
6. F Dereferencing a pointer gives you access to the memory location at that address. // only if it's a pointer to a pointer
7. F A shallow copy is sufficient for handling dynamically allocated member variables.
8. F structs are less useful than classes because you cannot create a constructor for structs. // only difference is that structs are public by default
9. F The return value of a class destructor is nullptr. // no return value
10. F If you declare an array of N objects the constructor for that object is called once to create that array. // called N times
11. T The following class definition for A requires the entire class definition for B in order to compile.

```
class A {
public:
     void foo( B b);
private:
     B* pb;
};
```

12. T The new keyword requests memory to be allocated by the operating system and returns an address to the allocated memory.
13. F With the declaration:

```
int* pInt = new int( 10 );
```

the correct way to delete the variable pInt is to say:

```
delete [] pInt;
```

Name: Noshir Janmohamed

14. F In the declaration:

```
class Foo : public Bar
```

Bar is the derived of the class Foo.

15. F Given the class declaration above and the objects:

```
Foo f;   Bar b;
```

it is always safe to say:

```
f = b; //only if there is an explicit assignment operator or no dynamically
```

16. T Stacks are considered to be First-in First-out abstract data types. // allocated memory.

17. T Eliminating errors at the pencil and paper stage makes it much easier to produce a correct program in the later steps of the problem-solving process.

18. T Template is a feature of C++ that allows us to write one code for different data types.

19. F You can declare a non-pointer instance of class that has pure virtual functions.

20. F When overloading the assignment operator for an object the return value is always a non-reference object type. // return val is a reference to object type

Multiple Choice: Choose the answer that fits best. Write out your letter choice into the provided space.

21. d Suppose the variable s is a pointer to a structure that has a member variable called address. Which of the following is the correct way to access that member variable:

a. s->address;
b. (*s).address;
c. s[0].address;
d. All of the above.
e. None mentioned.

*(s+i) == s[i]

22. d What is the lifetime of dynamically allocated data?

a. In the function it is defined.
b. Within the main function.
c. Within the class it is allocated.
d. Until it is deleted.
e. None mentioned.

Name: Nashir Jannohamed

23. b Given the following classes:

```
class Engine {
public:
    ...
};
```

```
class Car {
public:
    ...
private:
    Engine e;
};
```

```
class Toyota : public Car {
public:
    ...
};
```

the proper order of construction for the declaration `Toyota prius;` is:

- X a. Enter constructor for Toyota → Toyota constructor body → Car constructor → Car constructor body → Engine constructor.
- (b.) Enter constructor for Toyota → Car constructor → Engine constructor → Car constructor body → Toyota constructor body.
- X c. Enter constructor for Toyota → Engine constructor → Car constructor → Car constructor body → Toyota constructor body.
- X d. Enter constructor for Toyota → Toyota constructor body → Car constructor → Car constructor body → Engine constructor.
- e. None mentioned.

24. b A class with at least one pure virtual function is called:

- a. Abstract Data Type.
- (b.) Abstract Base Class.
- c. Polymorphic Template.
- d. Dynamically Allocated.
- e. None mentioned.

25. c Which keyword can be used in creating Templates?

- a. class.
- b. typename.
- (c.) both class and typename.
- d. function.
- e. None mentioned.

26. a. Which of the following is an advantage of singly linked lists over arrays?

- (a.) Insertion of elements into the middle
- X b. Accessing elements from the middle.
- X c. Insertion of elements at the end.
- X d. Easier to program.
- e. None mentioned

Name: Nashir Jammohamed

27. ___e___ Given the following class definitions, what is the resulting output:

```
class Robot {
public:
     virtual void dance() = 0;
     virtual void sing() = 0;
     void compute() { cout << "111 "; }
};
class HappyRobot :public Robot {
public:
     virtual void dance() { cout << "D "; }
     void compute() { cout << "222 "; }
};
void main() {
     Robot *r = new HappyRobot();
     r->dance();
     r->sing(); Never implemented
     r->compute();
}
```

- a. 0 0 111
- b. 0 0 222
- c. D 0 222
- d. D 0 111
- (e.) Compile error.

28. ___a___ Including the class definitions above with the one below, what is the resulting output?

```
class EcstaticRobot : public HappyRobot {
public:
     virtual void sing() { cout << "S "; };
     void compute() { cout << "333 "; }
};
void main() {
     Robot *r = new EcstaticRobot();
     r->dance();
     r->sing();
     r->compute();
}
```

- (a.) D S 111
- b. D S 222
- c. D S 333
- d. 0 0 111
- e. Compile error.

Name: Nashir Jammohamed

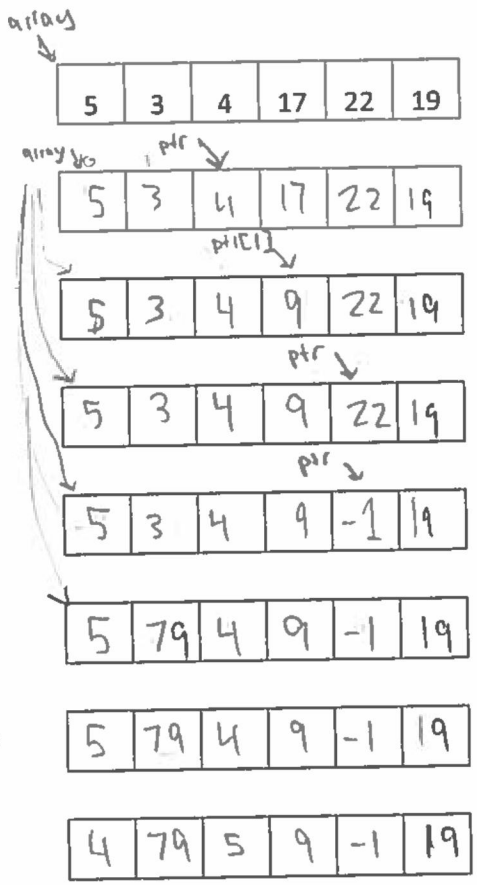
29. d A copy constructor for an object is called:
   a. Only if you explicitly define one.
   b. When making an assignment between two existing objects. //assignment operator
   c. When you want to make a shallow copy of an object.
   (d.) When creating a new object from an already existing object
   e. None mentioned

Short Answer:

30. a. What is the output of the following program? The cells are provided for your convenience.

```
void swap1(int* a, int *b) {
     int* temp = a;
     a = b;
     b = temp;
}
int main() {
     int array[6]
      = { 5, 3, 4, 17, 22, 19 };

     int *ptr = array + 2;

     ptr[1] = 9;

     ptr += 2;

     *ptr = -1;

     *(array + 1) = 79;

     swap1(&array[0], &array[1]);

     swap2(array, &array[2]);

     for (int i = 0; i < 6; i++)
          cout << array[i] << " ";
}
```

```
void swap2(int* a, int *b){
     int temp = *a;
     *a = *b;
     *b = temp;
}
```



Output:

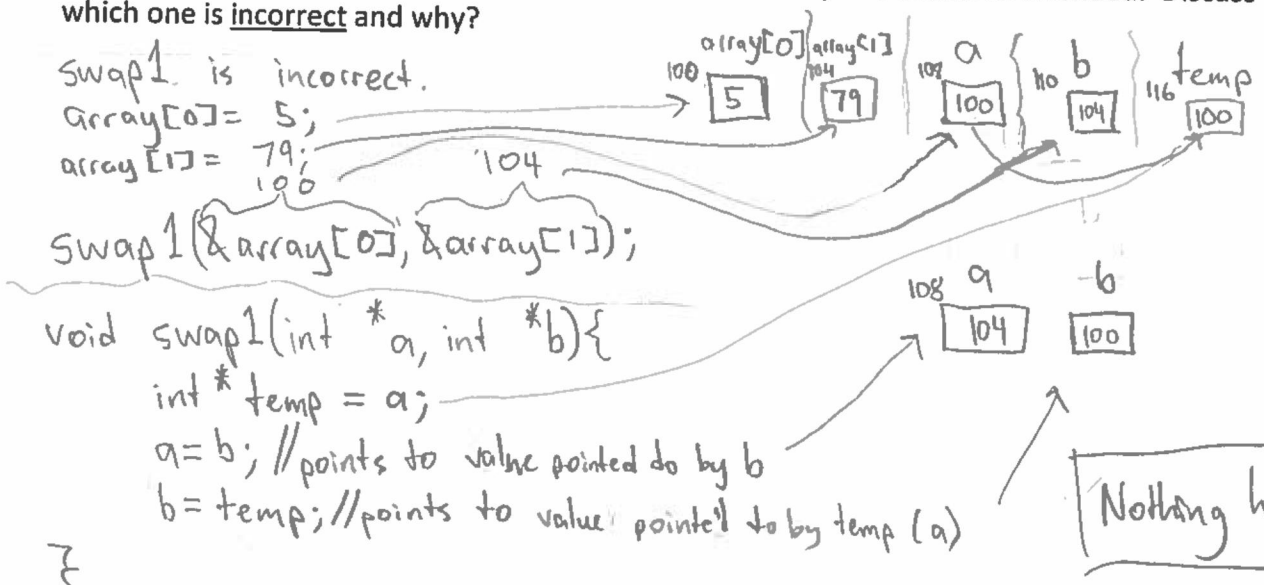
4 79 5 9 -1 19

Name:

b. In the previous part one of the swap functions did not swap the values as intended. Discuss which one is incorrect and why?

swap1 is incorrect.

```
array[0] = 5;
array[1] = 79;
```



c. For the correct swapping function redefine that function as a templated function, use the function identifier swap.

```
template <typename Type>
Void swap(Type* a, Type* b){
    Type temp = *a;
    *a = *b;
    *b = *temp;
}
```

Name: Nashir Jannohamed

31. a. Suppose you have the following class and it is passed into your templated swap function. What does the program print? (You can assume that the first parameter of the constructor is always greater than zero)

Output: 1) -1 -1 -1
2) 2 2

```
class Thing {
public:
   Thing(int x, int y):a(x),b(y) {
      p = new int*[a];//2 d array size a
      for (int i = 0; i < a; i++) {
               p[i] = new int(b);//new int(b) returns pointer of type int to an int w/
      }                          //value of b?
   }
   void print() {
      for (int i = 0; i < a; i++)
           cout << p[i][0] << " ";//prints content pointed to by p[i], essentially *(p[
      cout << endl;
   }
private:
   int a, b, **p;
};

int main() {
1) Thing *t1 = new Thing(2, 2),
2)       *t2 = new Thing(3, -1);
3) swap(t1, t2);
4) t1->print();
5) t2->print();
}
```

b. There are several (non-syntactical) issues with the program as is. State what the issues are and where they occur. (Note you just have to discuss, you don't have implement anything here.)

1) There is no overloaded, assignment operator, so there might be issues w/ the dynamically allocated number variable **p. There isn't in this case, but things could get messy.
2) There is no getter function for the derived classes!! :(
3) There is no destructor (for Thing), which will lead to a memory leak, as there is a dynamically allocated memory in each object.
4) t1 and t2 are not deleted before main ends which is a memory leak.
X 5) It is possible for someone to input a negative value for Thing.a in the constructor, which would lead to a runtime error.
I didn't read description carefully !!

Name: Nashir Janmohamed

c. Assuming it is declared in the class definition, implement a destructor for thing.

```
Thing::~Thing() {
    for (int i=0; i<a; i++){
        delete p[i];
    }
    delete [] p;

}
```

d. Assuming it is declared in the class definition, implement a copy constructor for Thing.

```
Thing::Thing(const Thing &other) {
    a = other.a;
    b = other.b;
    p = new int*[a];
    for (int i=0; i<a; i++){
        p[i] = new int(b);
    }

}
```

e. Assuming it is declared in the class definition, overload an assignment operator for Thing.

```
Thing & Thing::operator=(const Thing &other) {
    if (this == &other)
        return (*this);
    for (int i=0; i<a; i++){
        delete p[i];
    }
    delete [] p;
    a = other.a;
    b = other.b;
    p = new int*[a];
    for (int i=0; i<a; i++){
        p[i] = new int(b);
    }
    return (*this);
}
```

Name: Nashor Jamnohamed

32. a. Suppose have we a class that inherits from Thing called Mabob with definition as follows:

```
class Mabob : public Thing {
public:
     Mabob(int x, int y):Thing(x,y) {
          q = new int(y);
     }
     ~Mabob() { delete q; }
     void print() {
          for(int i = 0; i <= *q; i++)
               Thing::print();
     }
     void majig() {
          int *t = new int(2*(*q));
          delete q;
          q = t;
     }
private:
     int *q;
};
```

Assuming we have all the changes you've made in previous problem. What other changes to Thing should we make in order to use polymorphism? For example:

```
Thing *mb = new Mabob(2, 3);
mb->print();
delete mb;
```

We need to make print() virtual in Thing, and we also need to make Thing's destructor virtual. While not necessary, we should also prepend the virtual keyword to Mabob's implementation of print and to its destructor. It doesn't affect the program, but it is good style, and provides clarity to someone trying to understand the logic of the program (as well as to the programmer). We must also be careful about passing in negative values for y, as we could have an infinite loop if *q is <0.

Name: Nashir Janmohamed

b. Discuss the ways that Mabob exhibits the three properties of inheritance relative to Thing.

1) reuse	2) extension	3) Specialization
Mabob reuses the y value that is a member variable (b) of Thing. It also reuses Thing's print() function.	Mabob implements a new function, majig(), not found in Thing.	Mabob overrides (or should override) Thing's print() function.

c. Assuming it was declared in the class definition, overload the assignment operator for Mabob:

```
Mabob & Mabob::operator=(const Mabob &other) {
    if (this == &other)
        return (*this);
    Thing:: operator = (other);
    *q = *(other.q);

    return (*this);
}
```

Name: Nashir Janmohamed

33. What is the output of the following program?

```
class Paw {
public:
   Paw() {cout<<"P ";}
   ~Paw() {cout<<"~P ";}
};
```

```
class FluffyButt {
public:
   FluffyButt()  {cout<<"F ";}
   ~FluffyButt() {cout<<"~F ";}
};
```

```
class Doge {
public:
   Doge() {cout<<"D ";}
   ~Doge() {cout<<"~D ";}
   virtual void run() {cout<<"R ";}
   void bark() {cout<<"B ";}
private:
   Paw m_paws[4];
};
```

```
class Corgi :public Doge {
public:
   Corgi() {cout<<"C ";}
   ~Corgi() {cout<<"~C ";}
   virtual void run() {cout<<"r ";}
   void bark() {cout<<"b ";}
private:
   FluffyButt m_butt;
};
```

```
void main() {
   Corgi Ein;
   cout << endl << "---" << endl;
   Ein.bark(); //not polymorphism, calls derived
   Ein.run(); //
   cout << endl << "---" << endl;
   Doge *Pluto = &Ein; //no constructor call, nothing created
   cout << endl << "---" << endl;
   Pluto->bark(); //not virtual calls Doge::bark();
   Pluto->run(); //virtual, calls derived
   cout << endl;
   cout << endl << "---" << endl;
}
```

Output

1) P P P P D F C // spaces in between every letter *
2) - - -
3) b r // *
4) - - -
5) // intentionally blank
6) - - -
7) B r // *
8) // intentionally blank
9) - - -
10) ~C ~F ~D ~P ~P ~P ~P //*

1) create Ein
   i) enter Corgi constructor
   ii) enter Doge constructor
   iii) construct m_paws 4 times
   iv) execute Doge constructor
   v) construct m_butt
   vi) execute Corgi constructor
2) destroy Ein // !! animal cruelty is no joke
   i) Corgi destructor
   ii) destroy m_butt
   iii) Doge destructor
   iv) destroy m_paws 4 times

Name: Nashir Janmohamed