Name _____ Login name _____

Section: _____ TA _____

General Instructions

- DO **NOT** PUT YOUR **SSN** ON ANYTHING!
- DO NOT WRITE YOUR NAME ON ANY PAGE EXCEPT THIS ONE!
- Turn off any noise making device, especially **CELL PHONES**. You may lose up to one letter grade if your device disturbs the peace of the exam.
- You have 50 minutes. Relax, **pace yourself,** and pay attention to the point values.
- The exam is programming and short answer.
- Do not add features that are not required by the problem. For example, if the instructions **don't say anything** about user input, then your program should **not** take user input. If you aren't sure, ask.
- Do problems you are confident about first. If you finish the problems you know, write what you do know about other problems to gain partial credit; but erroneous information may detract from that credit or irritate the grader, so don't make stuff up.
- Read *all* the directions *carefully* on each problem.
- Often writing a fast, rough version of a program in English or pseudocode will make your coding faster and more accurate. It also enables me to give partial credit in some circumstances.
- You may assume that input will not produce errors for the procedures described, unless the questions say otherwise.
- Do not do unnecessary testing. For example, testing for both `x < 0` and `x >= 0` instead of using one test and then `else` would be considered unnecessary testing.
- Have fun! Learn something.

Error definition

Errors do not always create problems in output, but should be considered errors anyway, as we do in class. This means errors include, but are not limited to:

- incorrect number of parameters
- writing to invalid memory
- reading from invalid memory
- other compilation errors
- other runtime errors

Errors do not include type coercion in arithmetic (for example, assigning a double to an int) unless that impairs the correctness of a program.

Examine the code below. For the following questions, answer as if the line of code in question were located at the commented line.

```
class A{
  public:
  char * c;
};

int main(){
  A a;
  // HERE
  ...
}
```

1. (3 pts) Explain the effect of replacing the commented line with strcpy(a.c, “spam”);

2. (3 pts) Assume c points to a C-string. How would you print the address of the string in memory?

3. (3 pts) Name the members of the Big Three that this class should probably implement.

4. (6 pts) There are two ways of writing a function that could modify the value of i in main(). Write the **prototype** and **call** for one method below.

_____

```
int main(){
    int i = 0;

    _____
    return 0;
}
```

5. (6 pts) Whichever method you chose for number 4, use the other method here.

_____

```
int main(){
    int i = 0;

    _____
    return 0;
}
```

6. (6 pts) Assume an instance of A (shown) is declared on the heap. Write the **definition** of a function that can take a pointer to that instance as a parameter, and have the function modify i.

```
struct A{
  int i;
};
```

Your definition may not extend below this line.

```
0     #include <iostream>
1
2     using namespace std;
3
4     class myNum{
5      private:
6         double value;
7      public:
8
9         //assume class has a constructor here that takes a double argument
10
11        const myNum operator+(const myNum& other);
12    };
13
14    int main(){
15
16        myNum a(7);
17        myNum b(8);
18
19        a = b;
20
21        a = a + 3;
22
23        a = 3 + a;
24
25        a = b + a;
26
27        return 0;
28
29   }
```

7. (4 pts) Which line in main() above will cause a compilation error? Why?

8. (8 pts) Suppose the operator+ function was **not** a member function, but still needed direct access to private variables of myNum. Write the function definition.

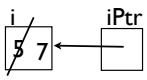
9. (18 pts) Fill in six blanks on this page after reading this and the next page too.

```
3     //READ ALL OF THIS PROBLEM (2 pages) BEFORE YOU BEGIN ANSWERING IT.
4     //DO NOT MODIFY ANY OF THE CODE YOU ARE GIVEN.
5     //ONLY WRITE CODE WHERE YOU ARE ASKED TO WRITE CODE.
6
7     class Record{
8     public:
9         char * name;
10        int age;
11
12        //Write a constructor prototype below. It will take no parameters.
13
14        _____________________________________________________________
15
16        //Write a constructor prototype below. It will take a char* parameter.
17
18        _____________________________________________________________
19        //Write an accessor function prototype for name below. Because
20        //this is an accessor, users of this function should not be able
21        //to modify name.
22
23        _____________________________________________________________
24
25        void changeName(char * newName);
26    };
27
28    //Changes the name of an instance of Record. This definition may not
29    //be well written, but that is ok for now.
30    void Record::changeName(char * newName){
31        name = newName;
32    }
33
34    int main(){
35
36        //Watch carefully for the words HEAP and STACK below.
37        int i = 5;
38        i = 7;
39        int *iPtr = &i;
40
41        //Declare an instance of Record on the stack.
42
43        _____________________________________________________________
44        //Declare an instance of Record on the heap using the second
45        //constructor. Write the constructor on the next page first!
46
47        _____________________________________________________________
48        //Give an age to the Record on the stack.
49
50        _____________________________________________________________
51        //Change the name of the Record on the heap using the changeName
52        //function defined above.
53
54        _____________________________________________________________
55
56        return 0;
57    }
```

10. (6 pts) Write a constructor definition here for the second constructor on the previous page. Constructor should allocate memory to hold a C-string.

11. (6 pts) Write an accessor function definition (see prototype on previous page) for name here. Because this is an accessor, users of this function should not be able to modify name.

This is an example drawing for lines 37 to 39 from the previous page. Changed values should be represented as here. Put line numbers in your drawing!

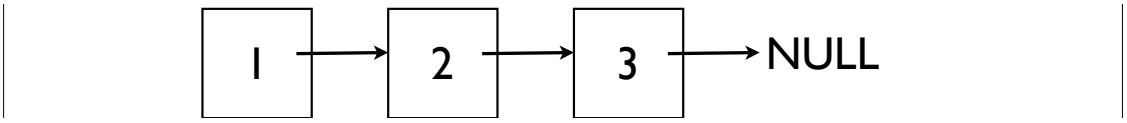


12. (6 pts) See the drawing instructions above. Draw the heap showing the declaration from line 47. Be sure to show the effect of your constructor from this page. Then show the effect of the call on line 54.

    Your drawing is most important here, so make it legible. You may add clarifying comments if you wish.

13. (6 pts) **Explain** what is fundamentally wrong with the definition for Record (not what is wrong with its member functions). What happens as a result? What should be done instead?

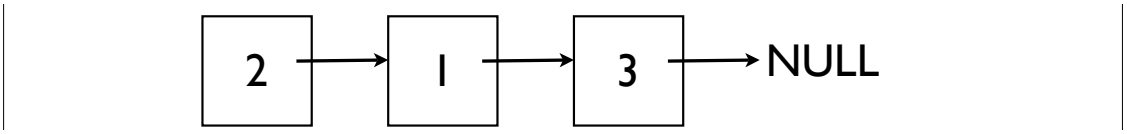
14. (8 pts) Using the definitions below, write code in main() to create the list drawn here. Make your own drawing to the side as you work to help yourself keep track. Do not use **any** functions that are not shown here. Do not complete main().




```
struct Node {
    int data;
    Node * next;
};

class LList {
 public:
    Node * head;
    LList() { head = NULL; }

    /* inserts at FRONT of list*/
    void insert(Node *n);
};

int main(){
```

15. (5 pts) Write a line of code that will print the value stored in the second node in the list.

16. (5 pts) Now, using the list's head, write code to make the list look like the drawing below.



17. (7 pts) Briefly explain how this code can cause an error at runtime. Use a drawing to clarify your text. Hint: the answer has nothing to do with the return value.

    The problem occurs in line 15. Carefully trace the code.

```
0     #include <iostream>
1
2     using namespace std;
3
4     class B{
5     public:
6         char * s;
7         B& operator=(const B& other);
8     };
9
10    int main(){
11
12        B b1;
13        b1.s = new char[30];
14        strcpy( b1.s, ``idempotent'');
15        b1 = b1;
16        return 0;
17    }
18
19    B& B::operator=(const B& other){
20        if (strlen(s) > strlen(other.s))
21            strcpy(s, other.s);
22        else{ //possibly not enough space, must re-allocate
23            delete[] s; //get rid of old space
24            s = new char[1 + strlen(other.s)];
25            strcpy(s, other.s);
26        }
27        return *this; //this line (at least) is correct ;) It just returns
28                      //the object that called this member fcn.
29    }
```

18. (3 pts) What is the actual difference between structs and classes?

19. (3 pts) What is the stylistic difference between how we use structs and classes?

20. (5 pts) Write a simple makefile that will separately compile two .cc files, then link the .o files if/when they are up to date. There are no .h files involved.

21. (5 pts) Write a short program that writes 5 random numbers to a text file named “spam”. It should write different numbers[1] each time it is called.

[1] usually