Midterm Exam

Name: ________________________________  Soc.Sec#: ________________

The following code applies to questions 1-5. Suppose the proper include files are used. Sections of code marked [.X.], where X is a number are to be decided by you in the questions.

class Cable {
    int city1, city2, cost;
public:
    Cable (int c1, int c2, int c) { city1 = c1; city2 = c2; cost = c; }
    int getCost () { return cost; }
};

// costCompare not shown

ostream & operator<<(ostream &out, Cable cable) {
    out << "[" << cable.getCost() << "] ";
    return out;
}

int main() {
    int city1, city2, cost, count=0;
    Cable *cables[100];

    for (int i=0 ; i < 100 ; i++) cables[i] = [.2.]
    qsort(cables, 100, sizeof(Cable*), costCompare);
    for (int i=0 ; i < 100 ; i++) cout << [.1.]cables[i];
    cout << "\n";
}

1. What should be substituted for [.1.]?
   
   {(a)} * (b) & (c) -> (d) nothing

2. What can be substituted for [.2.]?
   
   (a) Cable{i, i+1, rand() % 200};
   (b) new Cable();
   {c}) new Cable(i, i+1, rand() % 200);
   (d) Cable(i, i+1, rand() % 200);
3. Which of the following can be in costCompare?

   (a) if (((Cable **)x).getCost() < (**(Cable **)y).getCost()) return -1;
   (b) if ((*(Cable *)x).getCost() < (*(Cable *)y).getCost()) return -1;
   (c) if ((*(Cable *)x)->getCost() < (*(Cable *)y)->getCost()) return -1;
   (d) if (((Cable *)x).getCost() < (**(Cable *)y).getCost()) return -1;

4. What should be substituted for [.4.}? (a) *Cable (b) Cable (c) &Cable (d) Cable

5. What is missing from class Cable?

   (a) friend int costCompare(const void*, const void*);
   (b) friend ostream &operator<<(ostream &, Cable);
   (c) this->city1 = c1; this->city2 = c2; this->cost = cost;
   {d]} Nothing

6. Which of the following can be regarded as an abstraction?

   (a) A linked list (b) A list (c) A topological sort (d) A struct

7. Why do we sometimes want to pass a pointer to a function to a function?

   (a) To avoid casting.
   {(b)} So the function can be used with a variety of objects.
   (c) To reduce execution time in some contexts.
   (d) So the function can return a void value.

8. Why is recursion so useful?

   (a) A recursive program can be transformed to a dynamic program.
   (b) There is sometimes no alternative to recursion.
   {c]} It can result in short code whose correctness is more easily checked.
   (d) It can result in faster code that uses less space.

9. What operation is always most efficient for a singly linked list?

   (a) Displaying a named object in the list.
   (b) Retrieving an object from the list.
   {c]} Inserting an object at the front of the list.
   (d) Inserting an object at the rear of the list.

10. When removing an object from a doubly linked list...

    (a) an object is deleted.
    (b) the pointer used to locate the object is one ahead of the current object.
    (c) the pointer used to locate the object is one behind the current object.
    (d) a pointer must be set to the object to be removed.
    {e]} none of the above.
11. What is printed by the following code:

```cpp
class Object {
public:
    int number;
    Object(int number) { this->number = number; }
};

int main() {
    Object *object = new Object(10);
    delete object;
    int number = *new int(13);
    Object *c_object = new Object(15);
    cout << object->number << "\n";
}
```

(a) 10 (b) 13 (c) 15 (d) Segmentation fault

12. What is true about the following code:

```cpp
class Object {
public:
    int number;
    Object() { number = 10; }
};

int main() {
    Object *object = NULL;
    cout << object->number << "\n";
}
```

(a) Segmentation fault at runtime
(b) Prints the number 10
(c) Does not compile
(d) Prints 0

13. Suppose 1000 integers are stored in an array of ints in increasing order. Locating the array element containing a particular given integer takes no more than how many comparisons ("if" tests)? We are looking for the smallest number that applies.

(a) 10 (b) 20 (c) 30 (d) 1000
14. What is wrong with the following:

```c
long ham (int n) {
    if (n == 1) return 1;
    long p = ham(n-1)*(1+ham(n-2));
    return p;
}
```

Needs another boundary condition

15. What is wrong with the following:

```c
long ham (int n) {
    if (n == 1) return 1;
    return = ham(n-1)*ham(n+1);
}
```

Will stack indefinitely

16. Given

```c
class Object {
    public:
        int number;
        void doit () {
            (this+1)->number = number+1;
        }
};
```

How and in what context is `doit` probably intended to be used?

There is an array of Objects and invocation of `doit` causes number of the next

Object in succession to be incremented
17. What is wrong with the following:

```cpp
class Object {
    int number;
    char *name;
    int fact(int n) { if (n==1) return 1; return n*fact(n-1); }

public:
    Object(char *name, int number) {
        this->number = number;
        this->name = name;
    }
    char *getName () { return name; }
    int factorial () { return fact(number); }
};
```

Not making a copy of the string “name”

18. What is wrong with the following:

```cpp
...  
Pickle *pickle;  
theboss.Read();  
while (pickle == theboss.getNextPickle()) {
    pickle->juicy();
}
...
```

Should be pickle = theboss.getNextPickle()