CPSC 427 Midterm;

1. (5 points) What is one advantage (in terms of safety) to using a reference when passing something to a function verses passing as a pointer?

A reference can never be null (0). So you do not have to check it for null like you do a pointer.

2. (5 points) What does it mean if your program builds correctly in Eclipse (that is it generates an executable) but still shows errors in the code? What do you do about it?

Indexer problem. Refresh all files and rebuild the index

3. (5 points, multiple choice) Where does vector.end() point? What happens if you dereference it?

The last entry in the vector, you get the last entry if you dereference it.

One past the last entry in the vector, undefined behavior if you dereference it.

The last entry in the vector, undefined behavior if you dereference it.

One past the last entry in the vector, dereference safely returns the last entry

Q3
-2 1st
-2 3nd
-3 4th

4. (5 points, multiple choice) For the following code. What does the preprocessor do and what is the preprocessors output?

```
//myfunc.cpp
#include "myfunc.h"
std::string myfunc()
{
   return "hello world";
}
```

Inserts contents of myfunc.h into myfunc.cpp. The output is pure C++

Inserts only the called portions of myfunc.h into myfunc.cpp. The output is pure C++

Inserts std::myfunc() into myfunc.cpp, The output is pure C++

There is a preprocessor error, so nothing happens

Q4
-2 2nd
-2 3nd
-3 4th

5. (5 points, multiple choice) For the following code. What does the compiler do and what is the compiler output?

```
//myfunc.cpp
#include "myfunc.h"
std::string myfunc()
{
   return "hello world";
}
```

Nothing, there is a preprocessor error

Takes the pure C++ and compiles it to myfunc.o

Generates myfunc.exe from the pure C++

Prints the message "hello world" to the console

Q5
-2 1st
-3 3nd
-3 4th

6. (6 points, multiple choice) What is the difference between #include "myfile.h" and #include <myfile>

No real difference, you can use either. The compiler will find it.

The one with <> is a predefined system file, the compiler looks wherever predefined system files are kept, the one enclosed in "" appears to be a user file, the compiler will look in the users directory.

The one in <> is used for standard library headers so you cant put myfile.h in it. "" is for user files.

If the header file has a .h extension you put it in "". If not it goes in <>.

-3 1st and 4th
-0 3rd wrong but understandable

7. (7 points, multiple choice) For the following function , I expect the function to return myobject to me but not necessarily the same object I passed in.

What goes in the ?1 and ?2 spots?

```
void myFunc( ?1 myType   ?2 myobject);
```

?1= ?2=
Nothing goes in either

?1= ?2= &
Nothing goes in ?1

?1= ?2= *
Nothing goes in ?1

Q7
-4 1st
-3 2nd

8. (7 points, multiple answer) What should go in a header file?

Include guards

Minimal number of includes so that the header compiles cleanly

variables

.cpp files

{} after each function definition

All function declarations from the corresponding cpp file that you want exposed to other files

Q8
-1 for all wrong
Except variables that's
-2 unless qualified
with const variables

9. (3 points) How do you resize arrays in C++?

You can't they are sized at compile time. You could if you are using dynamic memory though.

10. (5 points)Please define an enum called fruit that contains Oranges, Apples and Pears with Oranges=1.

```
enum fruit{Oranges = 1,Apples,Pears}
```

Q10
-1 forget oranges = 1

11. (5 points) What are the advantages to defining a variable with the fruit enum type as opposed to an int?

That variable can only be 1 of the 3 values (Oranges, Apples, Pears} and no other. If you use an int you can not restrict the assignment to 1 of those 3, any int will do, for ex

```
int apples=1;
int oranges=2;
int pears=3;

int myFruit=5; //what fruit is this? Cannot happen with enum
```

12. (6 points) Create an efficient struct called fruitinfo with the following fields
Fruit Type (can only be Oranges, Apples or Pears)
Number pieces
Time entered (use clock_t)

```
struct fruitInfo{
	fruit type;
	int numPieces;
	clock_t timein;
};
```

Q12
--2 don't use clock_t
-2 define enum and not a var of type fruit

13. (3 points) Create a vector called myVector that holds fruitinfo structs.

```
vector<fruitInfo> myVector;
```

14. (13 points) Create a method that adds a fruit, and number of pieces of fruit to the vector myVector. The function also sets the time that the fruitinfo struct was added to the vector. Here is a function declaration to get you started;
void addFruit(fruit myFruitType, int numPieces)

```
void addFruit(fruit myFruitType, int numPieces ){
	fruitInfo myFruitInfo;
	myFruitInfo.type = myFruitType;
	myFruitInfo.numPieces = numPieces;
	myFruitInfo.timein = clock();

	myVector.push_back(myFruitInfo);
}
```

Q14
-2 don't use clock()
-6 don't create fruitinfo struct
-6 don't push_back
Max of -10

15. (15 points) Write a function that will save only the top X most recent fruitinfo structs in the vector where X is a user determined number. For instance if I wanted only the 20 most recently entered entries the function would delete all but the latest 20. Here is the function declaration

```
bool myfunction (const fruitInfo &i,const fruitInfo &j)
{ return (i.timein>j.timein); }

void clearAllButLatest(int myNumb) {
        //are there enough entries
        int size = myVector.size();
        if (myNumb >= size)
              return;

        //sort by time
        std::partial_sort (myVector.begin(), myVector.begin()+myNumb,
                  myVector.end(),myfunction);

        std::vector<fruitInfo>::iterator itr(myVector.begin());

        //jump iterator ahead to where we want to start deleting
        itr +=myNumb;

        while (itr != myVector.end()){
                  itr = myVector.erase(itr);//erase returns updated iter
                                            //pointing to next element
        }
}
```

Q15
-3 do not verify that vector.size()>myNumb
-5 do not sort by time
-5 do not delete oldest
-3-5 minor syntax errors
-5-8 major syntax errors

-10 max off if attempted

16. **(5 points) Is vector the best data structure for this application? Why or why not?**
If you clear a lot from the middle as my loop above does then vector is not the best, lots of copy and reallocation. List may be more appropriate