PENN STATE UNIVERSITY
Department of Economics

Econ 597D Sec 001 Computational Economics — Gallant
Sample Midterm Exam Questions — Fall 2015
In class on Oct 20, 2015

1. Write a C++ program and a makefile to build an executable from it that prints "Hello World" to the terminal. Write a C++ program that writes "Hello World" to the file `hello.out`.

2. Consider the C++ program

```
#include "libscl.h";
using namespace std;
using namespace scl;
int main(int argc, char** argp, char** envp)
{
  vector<string> arguments;
  vector<string> environment;
  char** strptr = argp;
  for (int i=0; i<argc; ++i) {
    arguments.push_back(*strptr++);
  }
  strptr = envp;
  while (*strptr) {
    environment.push_back(*strptr++);
  }
  vector<string>::size_type idx;
  for (idx = 0; idx < arguments.size(); ++idx) {
    cout << arguments[idx] << '\n';
  }
  for (idx = 0; idx < environment.size(); ++idx) {
    cout << environment[idx]<< '\n';
  }
  return 0;
}
```

   If this program is built and the executable is named `prog01`, then what will be printed if it is called as `prog01 How now brown cow`

3. Rewrite the `for` loops of the program in question 2 so that the vectors `arguments` and `environment` are traversed by an `iterator` instead of a `vector<string>::size_type`.

4. The object oriented programming style has four attributes that make it useful in solving complex economic problems. List and describe each.

5. The STL vector allows any object to be stored in an container indexed by integers. What indexes a STL map class?

6. What is the purpose of a statement like #include <iostream>?

7. Is the addition operator (+) left associative or right associative? Is the assignment operator (-) left associative or right associative?

8. What will the following six lines of code print?

```
int a = 0;
cout << ++a << '\n';
cout << a++ << '\n';
cout << a << '\n';
{ int a = 7; }
cout << a << '\n';
```

9. Define type, object, and variable. Give an example of code that produces each.

10. Define definition and declaration. Give an example of code that produces each.

11. What is a typedef? Give an example of one that involves an iterator for a STL map.

12. What is the difference between these two expressions a += b and a = a + b? Which executes faster if the compiler does no optimizaion? Why?

13. Consider the declaration std::map<std::string,int> phonebook. Suppose that statements after the declaration fill phonebook with key value pairs where the key is a person's name, last name first, and the value is the person's phone number. Write a for loop to print the key value pairs. Will the names appear in alphabetical order in this list? Consider the declaration std::map<int,std::string> criscross. Write a while loop that will fill this map with phonebook's value being criscross's key and phonebook's key being criscross's value. Write a do ... while loop to print criscross's key value pairs. Will this loop print the key in ascending order, descending order, or in some other order?

14. Consider the declaration std::map<std::string,int> phonebook Suppose that statements after the declaration fill phonebook with key value pairs where the key is a

person's name, last name first. Write a `for` loop that contains a `switch` statement within it to print the `key value` pairs for those person's whose last name begins with `R`.

15. C++ built in types come in five groups. What are they?

16. What is a `typedef`? Why are they used? Give an example of one.

17. Does `char* str="How now brown cow";` define a C string or a C++ string. What does `cout << str[2] << '\n';` print? What does `cout << str << '\n';` print?

18. Does `std::string str="How now brown cow";` define a C string or a C++ string. What does `cout << str[2] << '\n';` print? What does `cout << str << '\n';` print?

19. Assume data is read as follows `realmat A; vecread("scores.dat",A);`, where the first column of A is y and the remaining columns of A are X. Write a few more lines of code using `libscl` that will compute the regression of y on X and print the regression coefficients.

20. What is the BLAS?

21. How do a `vector`, a `list`, and a `map` from the Standard Template Library differ? What is each good for?

22. Consider `std::vector<std::string> V(5);`. To what element does `V.begin()` point? To what element does `V.end()` point? What is the type of `V.begin()`?

23. Suppose `int i = 5;` and `int j = 10;`. What will `float x = i/j;` contain? If your answer is not 0.5, write a line of code that will cause `x` to contain 0.5.

24. Explain what a file guard is. Give an example of one.

25. Describe each of the following, explain when each should be used, give a function declaration that illustrates the syntax of each.

    (a) Call by value.

(b) Call by reference.

(c) Call by const reference.

(d) Call using a pointer.

(e) Call using a const pointer.

26. Will the compiler allow these two function declarations to be in the same header file:

    `double f(double x);`

    `double f(int x);`

27. Will the compiler allow these two function declarations to be in the same header file:

    `double f(double x);`

    `int f(double x);`

28. Will the compiler allow these two function declarations to be member functions of the same class:

    `double f(double x);`

    `double& f(double x);`

29. Will the compiler allow these two function declarations to be member functions of the same class:

    `double f(double x) const;`

    `double& f(double x);`

30. If a declaration in the public part of a class is `double f(double x);`, and `c` is an instance of that class, is this statement `c.f(x)=y;` legal?

31. If a declaration in the public part of a class is `double& f(double x);`, and `c` is an instance of that class, is this statement `c.f(x)=y;` legal?

32. For each of the following, state whether the statement will or will not compile.

    (a) `REAL f(realmat b) {b(1,1)=5.0; return b(1,1);}`

    (b) `REAL f(realmat& b) {b(1,1)=5.0; return b(1,1);}`

(c) `REAL f(realmat* bptr) {(*bptr)(1,1)=5.0; return (*bptr)(1,1);}`

(d) `REAL f(const realmat& b) {b(1,1)=1.0; return b(1,1);}`

(e) `REAL f(const realmat& b) {realmat a=b; a(1,1)=1.0; return a(1,1);}`

(f) `REAL f(const realmat* bptr) {realmat* a=bptr; return (*a)(1,1);}`

(g) `REAL f(const realmat* bptr) {const realmat* a=bptr; return (*a)(1,1);}`

(h) `REAL f(const realmat* bptr) {realmat a=*bptr; a(1,1)=1; return a(1,1);}`

(i) `REAL f(realmat b) {cout << "#1 "; return b[1];}`

(j) `REAL f(realmat* bptr) {cout << "#2 "; return (*bptr)[1];}`

(k) `REAL f(REAL b) {cout << "#3 "; return b;}`

(l) `REAL f(INTEGER b) {cout << "#4 "; return b;}`

33. Is the following code likely to cause a program to crash? If so, state why.

```
using namespace std;
vector<int>& f(vector<int> v, int n) {v.resize(n+1,0); return v;}
int main() { vector<int> u; u=f(u,5); cout << u[4] << '\n'; return 0; }
```

34. Is the following code likely to cause a program to crash? If so, state why.

```
using namespace std;
vector<int>& f(vector<int>& v, int n) {v.resize(n+1,0); return v;}
int main() { vector<int> u; u=f(u,5); cout << u[4] << '\n'; return 0; }
```

35. Is the following code likely to cause a program to crash? If so, state why.

```
using namespace std;
vector<int>& f(vector<int>* v, int n) {v->resize(n+1,0); return *v;}
int main() { vector<int> u; u=f(&u,5); cout << u[4] << '\n'; return 0; }
```

36. In the following code, to what element of `a` does `t` point at the beginning of the loop? To what element of `a` does `t` point at the end of the loop?

```
int n=5000;
double a[n];
\\ fill a with something
double* t = a;
double* top = a + n;
double sum = 0.0;
while(t<top) {
  sum += *t++;
}
```

37. What is the difference between a class and a struct?

38. Which of these while loops will execute faster? Why?

(a)
```
std::list<double> lst(1000000);
\\ ...
std::list<double>::iterator iter=lst.begin();
while(iter != lst.end()) {
  if (*iter > 0) {
    iter = lst.erase(iter);
  }
  else {
    ++iter;
  }
}
```

(b)
```
std::vector<double> vec(1000000);
\\ ...
std::vector<double>::iterator iter=vec.begin();
while(iter != vec.end()) {
  if (*iter > 0) {
    iter = vec.erase(iter);
```

```
    }
    else {
      ++iter;
    }
  }
```

39. Given the definition `std::vector<scl::realmat> vec(1000);`, to what elements of `vec` do the following iterators point?

    (a) `vec.begin()`

    (b) `vec.end()`

    (c) `vec.rbegin()`

    (d) `vec.rend()`

40. Consider the associative map

    `std::map<std::string,int> phonebook`,

    (a) What is the type of the `key` and what is the type of the `value`?

    (b) Given the iterator

    `std::map<std::string, int>::const_iterator itr=phonebook.begin();`

    write code that will print the `key` and the `value` of the element of the map pointed to by `itr`.

    (c) If "Jane" is not in `phonebook`, what will be printed and what will be the state of `phonebook` after the statement

    `std::cout << phonebook["Jane"] << '\n';`

    is executed?

    (d) Write code that will print `"Jane"`'s phone number if she is in `phonebook`, will not change `phonebook`, and will write a warning message if `"Jane"` is not in `phonebook`.

    (e) Because there can be more than one `"Jane"`, what would have been a better container class to use than `std::map`?

41. Write a recursive function that will generate all multi indexes of dimension `d` up to order `deg`.

42. Write a generic function that will compute the mean of a vector containing any arithmetic type.

43. What is a constructor? What is a destructor?

44. Give three examples where the default constructor is called.

45. Write code that will allocate an array of size 1000 on the heap. Write code that will delete this allocated space.

46. Why is a constructor put in the return statement.

47. If `container` is a typical container class upon which arithmetic operations can be defined and `A`, `B`, and `C` are instances of that class:

    (a) Should the addition operator for use in an expression such as `C = A + B` be defined as a friend function or a member function?

    (b) Should the addition operator for use in an expression such as `C += A` be defined as a friend function or a member function?

48. Consider

```
class container {
private:
  double* x;
  size_t len;
public:
  container(size_t sz) : len(sz) {x = new double[len];}
  ~container{delete [] x;}
};
```

    (a) Let `A` be an instance of that container. Define a member function that implements the bracket operator and can be used in the statement `A[i] = 5.0`.

    (b) Let the function `f` be declared as `void f(const container& A)`. Define a member function that implements the bracket operator and can be used in the statement `double x = A[i]` within the body of the function `f`.

49. Consider

```
class container {
private:
  double* x;
  size_t len;
public:
  container(size_t sz) : len(sz) {x = new double[len];}
  ~container{delete [] x;}
};
```

(a) Write a default constructor for this class.

(b) Write a copy constructor for this class.

(c) Write an assignment operator for this class.

(d) Implement the `+=` operator for this class.

50. Consider the following header:

```
#ifndef __FILE_NL_LEAST_SQUARES_H_SEEN__
#define __FILE_NL_LEAST_SQUARES_H_SEEN__

#include "libscl.h"

class model_base {
public:
  virtual scl::realmat f(const scl::realmat theta) = 0; //returns ehat=f(theta)
  virtual INTEGER get_n() = 0;  //returns the dimension of ehat
  virtual INTEGER get_p() = 0;  //returns the dimension of theta
  virtual ~model_base() { };
};

scl::realmat fit(const model_base& model);  // returns estimated theta

#endif
```

(a) Is `f` in class `model_base` a virtual function or a pure virtual function?

(b) Define a class that inherits from `model_base` that will implement the model

$$\hat{e}_i = y_i - \theta_1 - \exp(\theta_2\, x_i).$$

That class will need a constructor that can store the values $y_i, x_i, i = 1, \ldots, n$.

(c) Write a main that instantiates your class `model`, calls `fit`, and prints $\hat{\theta}$. Note, you do not have to write the function `fit`, just call it.

51. Use OpenMP to parallelize the following program.

```
#include "libscl.h"

using namespace std;
using namespace scl;

int main(int argc, char** argp, char** envp)
{
  const INTEGER arows = 100;
  const INTEGER acols = 200;
  const INTEGER brows = acols;
  const INTEGER bcols = 300;

  realmat a(arows,acols);
  realmat b(brows,bcols);
  for (INTEGER i=1; i<=a.size(); ++i) a[i] = cos(i) + log(i);
  for (INTEGER i=1; i<=b.size(); ++i) b[i] = sin(i) + i/b.size();
  realmat r(arows,bcols,0.0);

  for (INTEGER j=1; j<=bcols; ++j) {
    for (INTEGER k=1; k<=acols; ++k) {
      for (INTEGER i=1; i<=arows; ++i) {
        r(i,j) += a(i,k)*b(k,j);
      }
    }
  }

  std::cout << a << b << r << '\n';
  return 0;
}
```

52. When this executes

```
int a;
int b = 1;
a = ++b;
```

What will be the value of `a`?

What will be the value of `b`?

When this executes

```
int a;
int b = 1;
a = b++;
```

What will be the value of `a`?

What will be the value of `b`?

When this executes

```
#include <iostream>
int main()
{
```

```
  const char* hw = "Hello World";
  const char* s = hw;
  while (*s) std::cout << *s++;
  std::cout << '\n';
  std::cout << "x"<< *s << "x" << '\n';
  return 0;
}
```

What will be printed?

What will be the value of `*s` before the return statement?

Make it clear in your answer whether the last line printed will be `xx`, `x x`, or something else.