INSTRUCTIONS:

(1) DO NOT OPEN YOUR EXAM BOOKLET UNTIL YOU HAVE BEEN TOLD TO BEGIN.

(2) The total for the exam is 100 points

(3) There are 9 pages with 32 problem; 15 multiple-choice, 15 short answer, and 2 free response.

(4) You may skip one multiple-choice and one short answer problem for a total of 5 points.

(5) Use the backside of the paper as your scratch paper. The backside is not graded.

(6) If you make a mistake, cross it out or erase it. Otherwise, it will be graded, for better or for worse.

(7) No electronic devices (e.g. phones, calculators, mp3 players, etc.) are allowed.

(8) You are allowed one sheet of notes on Letter paper or smaller.

(9) Write your name on the upper-right corner of each page of the exam.

(10) To receive full credit, you must write legibly and your answers must be clear. You may need to sketch out answers to less straightforward problems on the scratch side before you write on the front side.

(11) Do not leave classroom until you have handed in the exam.

(12) The exam lasts 110 minutes
(1) (2 pts) Which of the following are true about the * character in a C++ program?
   (a) In the expression 1 * 2 it represents multiplication
   (b) In the declaration long* p it represents dereferencing a pointer
   (c) In cout << *p it represents the creation of a pointer variable p.
   (d) All of the above
   (e) None of the above

(2) (2 pts) Which of the following are true about a lambda function?
   (a) It is a nameless function
   (b) They are often used in conjunction with STL algorithms
   (c) They have a capture list
   (d) All of the above
   (e) None of the above

(3) (2 pts) Which of the following are true about a function which is templated?
   (a) It is itself not a function, but a way to create a function
   (b) It contains the keyword template
   (c) It makes use of a template parameter to represent a calling type
   (d) All of the above
   (e) None of the above

(4) (2 pts) Which of the following are true about a function parameter which is both const and a reference?
   (a) A copy is made of that parameter when the function is invoked
   (b) You cannot change the parameter inside the function
   (c) It only works with pointer types
   (d) All of the above
   (e) None of the above

(5) (2 pts) What is the meaning of cin.clear()?
   (a) It removes all characters from the input stream
   (b) It undoes the last input operation
   (c) It clears any errors, resets cin to be “good”
   (d) It closes the cin stream
   (e) None of the above
(6) (2 pts) If \( v1 \) is a `vector<int>` object, which of the following are true about `vector<int>` 
\( v2 = v1; \)?
(a) \( v1 \) and \( v2 \) represent the same `vector`.
(b) Changes to \( v1 \) also change \( v2 \).
(c) Each item in \( v1 \) must be copied.
(d) All of the above
(e) None of the above

(7) (2 pts) The C++ container `map` is ordered. Which of the following are true about `map`?
(a) Order is based on the keys of the elements
(b) The key must respond to the `<` operator
(c) The key is const
(d) All of the above
(e) None of the above

(8) (2 pts) Which of the following are true about C++ exceptions?
(a) Potential elements that might throw are wrapped in a try block
(b) If an error is thrown, control moves to the catch block of the same error type
(c) An uncaught exception will halt the program
(d) All of the above
(e) None of the above

(9) (2 pts) Which is the type returned by a `map<long, string>`'s `insert`?
(a) `long`
(b) `long&`
(c) `string`
(d) `bool`
(e) None of the above
long fn1(vector<long> &v) {
    long result = 0;
    for (auto i = v.begin(); i != v.end(); ++i) { // line1
        result += *i;
    }
    v.push_back(result);
    return result;
}

int main() {
    vector<long> v{1, 2, 3};
    cout << fn2(v) << endl; // line 2
    cout << v.size() << endl; // line 3
    cout << v.back() << endl; // line4
}

(10) (2 pts) What type is i on Line 1?
(a) vector
(b) vector<long>
(c) vector<long>*
(d) vector<long>::iterator
(e) None of the above

(11) (2 pts) What is the output produced by Line 2?
(a) 1
(b) 3
(c) 6
(d) 12
(e) None of the above

(12) (2 pts) What is the output produced by Line 3?
(a) 0
(b) 1
(c) 2
(d) 3
(e) None of the above
(13) (2 pts) What is the output produced by Line 4?
(a) 1
(b) 3
(c) 6
(d) 12
(e) None of the above

```cpp
int fn2(vector<long> &v1, vector<long> &v2) {
    int result = 0;
    while (v1.size() >= v2.size()) {
        v2.push_back(v1.back());
        v1.pop_back();
        ++result;
    }
    return result;
}

int main() {
    vector<long> v1{1, 2, 3, 4, 5};
    vector<long> v2{6, 7, 8};
    cout << fn1(v1, v2) << endl; // line 1
    cout << v1.back() << ":" << v2.back() << endl; // line 2
}
```

(14) (2 pts) What is the output produced by Line 1?
(a) 0
(b) 1
(c) 2
(d) 3
(e) None of the above

(15) (2 pts) What is the output produced by Line 2?
(a) 3:4
(b) 3:5
(c) 5:8
(d) 4:7
(e) None of the above
string fn3 (const vector<string>& v, string s) {
    auto pos = find(v.begin(), v.end(), s);
    return (pos == v.end()) ? "nope" : *pos;
}

(16) (3 pts) What is the type of pos is fn3

(17) (3 pts) If vector<string> v = {"bill", "fred", "bill", "george"}, what is the result of calling fn3(v, "Bill")?

size_t fn4 (const vector<string>& v, string s) {
    auto pos = find(v.begin(), v.end(), s);
    if (pos != v.end())
        pos = find(pos + 1, v.end(), s);
    return pos - v.begin();
}

(18) (3 pts) If vector<string> v = {"bill", "fred", "bill", "george"}, what is the result of calling fn4(v, "bill")?

(19) (3 pts) If vector<string> v = {"bill", "fred", "bill", "george"}, what is the result of calling fn4(v, "george")?

bool sort_fn (const string& s1, const string& s2) {
    return s1.back() < s2.back();
}

void fn5 (vector<string>& v) {
    sort(v.begin(), v.end(), sort_fn);
}

(20) (3 pts) After executing vector<string> v = {"bill", "fred", "bill", "george"}; fn5(v);, what is the result of v.front()?

(21) (3 pts) After executing vector<string> v = {"bill", "fred", "bill", "george"}; fn5(v);, what is the result of v.back()?
long a_fn(long t, long e) {
    return e*e + t;
}

long fn6(vector<long>& v) {
    return accumulate(v.begin(), v.end(), 0, a_fn);
}

(22) (3 pts) If vector<long> v = {1, 2, 3, 4}, what is the result of calling fn6(v)?

(23) (3 pts) If vector<long> v = {3, 4, 1, 2}, what is the result of calling fn6(v)?

long fn7(const vector<vector<long>>& v) {
    long result = 0;
    for (auto r : v) { // line 1
        for (auto c : r) {
            if (c % 2)
                result += c;
        }
    }
    return result;
}

(24) (3 pts) What is the type of r on line 1 of fn7?

(25) (3 pts) If vector<vector<long>> v = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}, what is the result of calling fn7(v)?
vector<long> fn8(const vector<long>& v1, const vector<long>& v2) {
    vector<long> result;
    for (auto e : v1) {
        auto i = find(v2.begin(), v2.end(), e);  // line 1
        if (i != v2.end())
            result.push_back(e);
    }
    return result;
}

(26) (3 pts) What is the type of r on line 1 of fn8?

(27) (3 pts) If v1 = 0, 1, 2, 3 and v2 = 2, 3, 4, 5, what is the result of calling fn8(v1, v2)?

template <class T>
string fn9(const vector<T>& v) {
    ostringstream oss;
    for_each(v.begin(), v.end(),
        [&oss](auto x) { oss << x; });  // line 1
    return oss.str();
}

(28) (3 pts) What is the type of x on line 1 of fn9?

(29) (3 pts) If vector<long> v = {1, 2, 3, 4, 5}, what is the result of calling fn9(v)?

(30) (3 pts) If vector<char> v = {'a', 'b', 'c'}, what is the result of calling fn9(v)?
(31) (15 pts) Write a function `BoundingBoxWidth` that receives as input a constant `vector` of `pair<double, double>` objects that represent \((x, y)\) coordinates and returns the width of the smallest box that encloses all of the points. For example, the triangle represented by \(
\{\{-1, 1\}, \{2, 0\}, \{3, -2\}\}\) would have width 4. **You should not use a for or while loop.** Hint: The `minmax_element` function will give you a pair of iterators pointing to the minimum and maximum elements and pairs are sorted first on their \(x\) coordinate and breaking ties with the \(y\) coordinate.

(32) (15 pts) Write a function `CountLetters` that receives a constant `vector` of `string`s and determines the total number of letters. For example, if given the sequence \(
\{\"alfa\", \"bravo\", \"charlie\"\}\) it should return 16. **You should not use a for or while loop.** Hint: The `accumulate` function will be useful. You may write a helper function or lambda.