| Row | Seat |
| --- | --- |
| | |

Final Exam CSCI 135: Programming Design and Analysis

Hunter College, City University of New York

Final Exam Date and Time:16 December 2021, 11:30 – 1:30 PM

Exam Rules
- Show all your work. Your grade will be based on the work shown.
- The exam is closed book and closed notes.
- When taking the exam, you may have with you pens and pencils, and the cheat sheet provided.
- You may not use a computer, calculator, tablet, phone, earbuds, or other electronic device.
- Do not open this exam until instructed to do so.

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating on examinations, obtaining unfair advantage, and falsification of records and official documents) as serious offenses against the values of intellectual honesty. The College is committed to enforcing the CUNY Policy on Academic Integrity and will pursue cases of academic dishonesty according to the Hunter College Academic Integrity Procedures.

| I understand that all cases of academic dishonesty will be reported to the Dean of Students and will result in sanctions. |
| --- |
| Name: Answer |
| Emp ID: |
| Email: |
| Signature: |
| Initial: |

Initial:

1. Short answer questions (3-point each).

   (1) Declare that class Dog as a subclass of Animal and inherits its public members.

   ```
   class Dog : public Animal
   ```

   (2) Declare a vector of double numbers, call it **weights**. Initialize with 17.2, 36.1, 65.

   ```
   vector<double> weights = {17.2, 36.1, 65}; //c++ 11 or above
   Or
   vector<double> weights;
   weights.push_back(17.2);
   weight.push_back(36.1);
   weight.push_back(65);
   ```

   (3) Suppose int arr[] = {3, 2, 8, 9}; What is *arr + *(arr+2) ?

   *arr is arr[0], which is 3 in our example.
   *(arr + 2) is arr[2], which 8.
   *arr + *(arr + 2) is 11.

   (4) Write the **header** of a function foo, for two given integers m and n, if n is not 0 and m is a multiple of n, returns true, otherwise, return false.

   ```
   bool foo(int m, int n); //no implementation is not need
   //whether ; is followed or not is fine, no need to deduct point without or without ;
   ```

   (5) What is the possible values of ( 3 + rand() ) % 6 + 2?

   **This problem will not be graded.** Reason: 3 + rand() might overflow (ie, surpass the limit of int), so you will get a negative int when (3 + rand()) % 6.

Initial:

(6) Declare a struct called TV, which includes the following data members: size as an int and model as a string.

```
struct TV
{
    int size;
    string model;
}; //; cannot be omitted. But if students missed it, give a warning, do not deduct point
```

(7) What is output for the following code?

```
char numToLetter(int grade)
{
  char letter;
  if (grade >= 90)
    letter = 'A';
  else if (grade >= 80)
       letter = 'B';
     else if (grade >= 70)
          letter = 'C';
        else if (grade >= 60)
             letter = 'D';
           else letter = 'F';
  return letter;
}

int main()
{
  int grades[] = {36, 60, 89, 90, 100};
  int size = sizeof(grades) / sizeof(grades[0]);

  int value = 0;
  char letter;
  for (int i = 0; i < size; i++)
  {
    letter = numToLetter(grades[i]);
    if (letter != 'A' && letter != 'B')
      value++;
  }

  cout << value << endl;
  return 0;
}
```

Return number whose corresponding letter grade is neither A nor B. There are 36 and 60 that fit this category. So the output is 2.

Initial:

(8) Read the following code. What is the output?

```
class Computer {
public:
   Computer()
   {
     id = id_generator;
     id_generator++;
   }

   int get_id() const
   {
      return id;
   }
private:
   static int id_generator;
   int id;
};

int Computer::id_generator = 1;

int main()
{
   Computer first;
   Computer second;
   Computer third;

   cout << second.get_id();

   return 0;
}
```

2

(9) Declare and initialize a two-dimensional int array called arr with the first row 1, 2, the second row 3, 4, and the third row 5, 6.

```
int arr[][2] = { {1, 2}, {3, 4}, {5, 6} };
or
const int NUM_COLUMNS = 2;
int arr[][NUM_COLUMNS] = { {1, 2}, {3, 4}, {5, 6} };
```

Initial:

(10) What is the output for the following code?

```
for (int i = 1; i <= 2; i++)
{
   for (int j = 1; j <= 3; j++)
      cout << i + j << " ";

   cout << endl;
}
```

```
2 3 4
3 4 5
```

2. Fill in blanks (10 points)

(1) Write code for each requirement.

Declare an int variable called it size and initialize it to be 20. Create a one-dimensional dynamic allocated memory array, call it data, of ints whose capacity is size.

```
int size = 20;
int* data = new int[20];
```

Set each element of data to be a random int in [100, 200].

```
for (int i = 0; i < size; i++)
    data[i] = rand() % (200 – 100 +1) + 100;
```

Find out the difference between the maximum and the minimum of array data.

```
int min = data[0];
int max = data[0];
for (int i = 1; i < size; i++)
     if (data[i] < min)
        min = data[i];
     else if (data[i] > max) //without else is also ok, just a little low efficiency, but then loop body needs to be included in a pair of {} since it contains two separated if-statements.
              max = data[i];

int difference = max – min;
//print or return difference is optional.
```

Initial:

(2) Define a **recursive** function that takes an int, return its number of digits. For example, if input is 123, then return 3. If input is -2, then return 1. If input is -23, return 2.
Define function header. The function name is numDigits, the given parameter is num.

```
int numDigits(int num)
```

{

If num has only one digit (can be negative), return 1.

```
if (num / 10 == 0)
  //can also write as if (num < 10 && num >-10) or if (abs(num) < 10)
   return 1;
```

Now num has more than one digit. Write recursive code to find out number of digits of num. Hints: suppose num is 123, how to get 12? What is the relationship between number of digits of 123 and number of digits of 12? Similarly, how to get 1 from 12? What is the relationship between number of digits of 1 and number of digits of 12?

```
return numDigits(num / 10) + 1;
    //num / 10 equals num without the least significant digit
```

}

In main function, write code to print the number of digits of 123 applying numDigits function.

```
cout << numDigits(123);
```

Initial:

(3) Define a function foo, for a given array arr of ints and its size, return type is empty. Define the function header.

```
void foo(int* arr, int size) //can also write as void foo(int arr[], int size)
```

For each adjacent pair arr[i] and arr[i+1] in arr, if arr[i] equals arr[i+1], replace arr[i] by twice of arr[i] and set arr[i+1] to be 0 (students may have different implementations.)

```
//work from leftmost adjacent pair to
//rightmost adjacent pair.
for (int i = 0; i < size-1; i++)
    if (arr[i] == arr[i+1])
    {
        arr[i] *= 2;
        arr[i+1] = 0;
    }
```

```
//work from rightmost adjacent pair to
//leftmost adjacent pair.
for (int i = size-2; i >= 0; i--)
    if (arr[i] == arr[i+1])
    {
        arr[i] *= 2;
        arr[i+1] = 0;
    }
```

After apply foo on array {2, 2, 1, 1, 0}, what does array looks like?

Merge from leftmost adjacent pair to rightmost adjacent pair,

(1) Check first adjacent pair 2 and 2 in {2, 2, 1, 1, 0}. Since they are equal, replace arr[0] by 2 * 2 = 4 and replace arr[1] = 0. The array becomes {4, 0, 1, 1, 0}.
(2) Move to the next adjacent pair 0 and 1 in {4, 0, 1, 1, 0}. They are not equal, do nothing.
(3) Move to the next adjacent pair 1 and 1 in {4, 0, 1, 1, 0}. They are equal, replace arr[2] by 2 * 1 = 2 and replace arr[3] by 0. The array becomes {4, 0, 2, 0, 0}.
(4) Move to the next adjacent pair 0 and 0 in {4, 0, 2, 0, 0}, They are equal, but changing arr[3] by 2 * 0 still returns 0 and arr[4] is replace by 0. The array is not changed.

Conclusion: the array is {4, 0, 2, 0, 0} after the above repetition statement.

Merge from rightmost adjacent pair to leftmost adjacent pair.

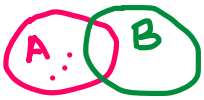
(1) Check rightmost adjacent pair 1 and 0 in {2, 2, 1, 1, 0}. Since they are not equal, do nothing.
(2) Check second rightmost adjacent pair 1 and 1 in {2, 2, 1, 1, 0}. They are not equal, replace arr[3] by 2 * 1 = 2 and arr[4] by 0. The array becomes {2, 2, 2, 0, 0}.
(3) Check third rightmost adjacent pair 2 and 1 in {2, 2, 2, 0, 0}. They are equal, replace arr[1] by 2 * 2 = 4 and replace arr[2] by 0. The array becomes {2, 4, 0, 0, 0}.
(4) Move to the next adjacent pair 2 and 4 in {2, 4, 0, 0, 0}, They are not equal, so the array keep it current value.

Conclusion: the array is {2, 4, 0, 0, 0} after the above repetition statement.

Initial:

3. (1) Define a function, for a vector of strings and a target string, find out whether the target string is in vector or not. If yes, return true, otherwise, return false.
   (2) Define function, for two vectors of strings vectA and vectB, find out all the strings that are in vectA but **not** in vectB, put them in a vector. Return that vector. For simplicity, we assume that no two elements in vectA are the same, neither is vectB. For example, if vectA is {"aaa", "bbb", "ccc", "ddd"} and vectB is {"ddd", "bb", "aaa"}, then the returned vector is {"bbb", "ccc"}.
   Hints: you may apply function in (1) when working the function in (2). You may need to use push_back and size methods of vector.

   Idea of the problem




   For each element in A, if it is not in B, then this element is in A but not in B. Put all these elements to vector result.

```
#include <iostream>
#include <vector>
using namespace std;
//Sample output:
//elements in vectors A but not in B
//bbb
//ccc

bool contains(vector<string> data, string target);
vector<string> commonElements(vector<string> vectA, vector<string> vectB);

int main()
{
    vector<string> vectA = {"aaa", "bbb", "ccc", "ddd"};
    vector<string> vectB = {"ddd", "bb", "aaa"};

    vector<string> result = commonElements(vectA, vectB);
    cout << "elements in vectors A but not in B" << endl;
    for (int i = 0; i < result.size(); i++)
        cout << result[i] << endl;

    return 0;
}

bool contains(vector<string> data, string target)
{
   for (int i = 0; i < data.size(); i++)
       if (data[i] == target)
          return true;

   return false;
}

vector<string> commonElements(vector<string> vectA, vector<string> vectB)
{
```

Initial:

```
    vector<string> result;
    for (int i = 0; i < vectA.size(); i++)
        if (!contains(vectB, vectA[i]))
           //condition (!contains(vectB, vectA[i])) can write as
           //(contains(vectB, vectA[i]) == false)
           result.push_back(vectA[i]);

    return result;
}
```

Initial:

4. Define class Square.
   (1) Data member is side, which is a number that may contain decimal numbers.
   (2) Define default constructor of class Square, set side to be 1.
   (3) Define non-default constructor of class Square which takes an input parameter side, if this given parameter is positive, use it to initialize data member side, otherwise, initialize data member side to be 1.
   (4) Define a method to reset data member side. If the given parameter is positive, then use it to reset data member side, otherwise, do not change data member side.
   (5) Define a method to get data member side.
   (6) Define a method to get the perimeter, which is four times side.

```
//g++ -c Square.cpp to avoid prompt that main function is not provided.
//-c means compilation only,there is no need to generate a runnable file.
//Only Square.o is generated after the above command.
class Square
{
public:
    Square();
    Square(double side);
    double getSide() const;
    void setSide(double side);
    double getPerimeter() const;
private:
    double side;
};

Square::Square()
{
    side = 1;
}

Square::Square(double side)
{
    if (side > 0)
       this->side = side;
    else this->side = 1;
}

double Square::getSide() const
{
    return side;
}

void Square::setSide(double side)
{
    if (side > 0)
       this->side = side; //this-> in left variable cannot omit
    //if (side <= 0), no need to use given parameter side
    //to update data member side.
}

double Square::getPerimeter() const
{
    return 4 * side;
}
```

Initial:

5. Define NUM_COLUMNS as a const with value 3. Define a method for a two-dimensional array of chars with NUM_COLUMNS columns, check whether there is a column with all space characters. For example, if we have
char arr[][NUM_COLUMNS] = { {'X', 'O', 'X'}, {'O', 'X', 'O'}, {'X', 'O', ' '}, {'O', ' ', 'X'} };
Illustrated as follows. Then the return would be false.

| | | |
|---|---|---|
| 'X' | 'O' | 'X' |
| 'O' | 'X' | 'O' |
| 'X' | 'O' | |
| 'O' | | 'X' |

Hints: for **each** column, count the number of spaces. If a column has all spaces, what is that number equal to?

Define NUM_COLUMNS as a const with value 3.

```
const int NUM_COLUMNS = 3;
```

bool column_all_space(char arr[][NUM_COLUMNS], int numRows)
{ //Your code goes here.

```
    int num;
    for (int j = 0; j < NUM_COLUMNS; j++)
    {
        num = 0;
        for (int i = 0; i < numRows; i++)
            if (arr[i][j] == ' ')
               num++;

        if (num == numRows)
           return true;
    }

    return false;
```

}

For completeness, I add main function, but it is optional.

```
int main()
{
    char arr[][NUM_COLUMNS] =
        { {'X', 'O', 'X'}, {'X', 'X', 'O'}, {'X', 'O', 'X'},
          {'X', 'O', ' '} };
    int numRows = sizeof(arr) / (sizeof(char) * NUM_COLUMNS);
    cout << column_all_spaces(arr, numRows) << endl;
    return 0;
}
```

Improvement: define function to see whether a column in arr contains target only.
bool column_same_value(char arr[][NUM_COLUMNS], int numRows, char target)