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 Key
Emp ID:
Email:
Signature:
Initial:

Initial:

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

   (1) Declare class Undergraduate as a subclass of Student and inherits its public members.

   ```
   class Undergraduate : public Student
   ```

   (2) Declare a vector of ints, call it **ages**. Initialize with 17, 36, 65.

   ```
   vector<int> ages = {17, 36, 65};
   Or
   vector<int> ages;
   ages.push_back(17);
   ages.push_back(36);
   ages.push_back(65);
   ```

   (3) Suppose int arr[] = {2, 3, 4}; What is *arr + *(arr+1) ?

   *arr is arr[0], which is 2 and *(arr+1) is arr[1], which is 3.
   Answer: 5

   (4) Write the **header** of a function foo, for given string **str** and an int representing **index**, if index is valid and the character at the index in str is a smaller letter, return true, otherwise, return false.

   ```
   bool foo(string str, int index);
   ```

   (5) What is the possible values of (1 + rand()) % 12 + 3 ?

   This problem will not be graded. Reason: 1 + rand() might overflow, ie, surpass the maximum value of int when rand() returns RAND_MAX. So (1 + rand())% 12 might get a negative int.

   (6) Declare a **struct** called Person, which includes the following data members: name as a string and age as an int.

Initial:

```
struct Person
{
    string name;
    int age;
}; //do not forget ;
```

(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[] = {20, 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;
}
```

Value is the number of grades that is either A or B, in grades array, 89, 90, 100 will get a letter grade A or B. So print out 3.

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

```
class Computer {
```

Initial:

```
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;

   cout << second.get_id();

   return 0;
}
```

The id for the first computer is 1, and the id for the second computer is 2. So print out 2.

(9) Declare and initialize a two-dimensional int array called arr with two row 1, 2, 3, 4, 5, the second row 6, 7, 8, 9, 10.

```
int arr[][5] = {{1, 2, 3, 4, 5}, {6, 7, 8, 9, 10}};
Or
const int NUM_COLUMNS = 5;
int arr[][NUM_COLUMNS] = {{1, 2, 3, 4, 5}, {6, 7, 8, 9, 10}};
```

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;
}
```

```
1 2 3
2 4 6
```

2. Fill in blanks (10 points)

(1) Write code for each requirement.

Declare an int variable called it size and initialize it to be 10. Create a one-dimensional dynamic allocated memory array, call it data, of ints whose capacity is size.

```
int size = 10;
int* data = new int[size]; //warning: cannot write as int data[] = new int[size];
```

Set each element of data to be a random int in [60, 100].

```
for (int i = 0; i < size; i++)
    data[i] = rand() % (100 – 60 + 1) + 60;
```

Write code to find out the average of array data.

```
int sum = 0;
for (int i = 0; i < size; i++)
   sum += data[i];

//calculate average
double average = (double) sum / size;
    //also can write as double average = sum * 1.0 / size; or
   //double average = (0.0 + sum) / size; or
   //any valid code to change either sum or size to be double type.
   //Suppose sum is declared as double type, then there is no need to put (double)
   //before it.
cout << "average = " << average << endl; //optional. cout << (double) sum / size; is OK
//no need to return average.
//Average may contain decimals, so it must be double type.
```

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.
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 && num < 10) //Or if (abs(num) < 10), or if (num / 10 == 0)
   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? How to get 1 from 12? What is the relationship between number of digits of 12 and number of digit of 1?

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

}

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

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

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); //or 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], set arr[i] to be zero and replace arr[i+1] by twice of arr[i+1]. (students may have different implementations)

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

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

After applying foo on array {2, 2, 1, 1, 0}, what does array look like?

When merge from leftmost adjacent pair to rightmost adjacent pair,

(1) The first adjacent pair share the same value, {2, 2, 1, 1, 0}, set arr[0] to be 0 and double the value of arr[1], we get {0, 4, 1, 1, 0}.
(2) The second adjacent pair 4 and 1 in array {0, 4, 1, 1, 0} do not have the same value, do nothing.
(3) The third adjacent pair 1 and 1 in array {0, 4, 1, 1, 0} have the same value, set arr[2] to be 0 and double the value of arr[3]. The array becomes {0, 4, 0, 2, 0}.
(4) The fourth (also the last) adjacent pair 2 and 0 in array {0, 4, 0, 2, 0} do not have the same value, do nothing.

Conclusion: the array becomes {0, 4, 0, 2, 0}.

When merge from rightmost adjacent pair to leftmost adjacent pair,

(1) The rightmost adjacent 1 and 0 do not share the same value, {2, 2, 1, 1, 0}, do nothing.
(2) The second rightmost adjacent pair 1 and 1 in array {2, 2, 1, 1, 0} have the same value, replace arr[2] by 0 and arr[3] by 2. The array changes to {2, 2, 0, 2, 0}.
(3) The third rightmost adjacent pair 2 and 0 in array {2, 2, 0, 2, 0} do not have the same value, do nothing.
(4) The leftmost adjacent pair 2 and 2 in array {2, 2, 0, 2, 0} share the same value, arr[0] changes to 0 and arr[1] is changed to 4. The array becomes {0, 4, 0, 2, 0}.

Conclusion: the array becomes {0, 4, 0, 2, 0}.

It happens in this example, whether we merge from leftmost or rightmost adjacent pair get the same value, this might not be true for a different instance, for example, {1, 1, 1}. When merge from left to right, we get {0, 2, 1}. When merge from right to left, we get {1, 0, 2}.

Initial:

3. (1) Define a function, for a vector of strings and a target string, find out whether the target string is in this 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 common in vectA and 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 vector with common elements is {"aaa", "ddd"}.
   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 of A, check whether it is in B or not. If it is in B, then that element is common to A and B.

```
#include <iostream>
#include <vector>
using namespace std;
//Sample output:
//common elements of vectors A and B are
//aaa
//ddd

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 << "common elements of vectors A and B are" << 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;
}
```

Initial:

```
vector<string> commonElements(vector<string> vectA, vector<string> vectB)
{
    vector<string> result;
    for (int i = 0; i < vectA.size(); i++)
        if (contains(vectB, vectA[i]))
           result.push_back(vectA[i]);

    return result;
}
```

Initial:

4. Define class Circle.
   (1) Data member is radius, which is a number that may contain decimal numbers.
   (2) Define default constructor of class Circle, set radius to be 1.
   (3) Define non-default constructor of class Circle which takes an input parameter radius, if this given parameter is positive, use it to initialize data member radius, otherwise, initialize data member radius to be 1.
   (4) Define a method to reset data member radius. If the given parameter is positive, then use it to reset data member radius, otherwise, do not change the radius of the current object.
   (5) Define a method to get data member radius.
   (6) Define a method to get the area of a circle. The formula is πr^2. To use π, you may use M_PI, which is define in cmath library.

```
#include <iostream>
#include <cmath>
using namespace std;

class Circle
{
public:
    Circle();
    Circle(double radius);
    double getRadius() const;
    void setRadius(double radius);
    double getArea() const;

private:
    double radius;
};

Circle::Circle()
{
    radius = 1;
}

Circle::Circle(double radius)
{
    if (radius > 0)
       this->radius = radius;
    else this->radius = 1;
}

double Circle::getRadius() const
{
    return radius;
}

void Circle::setRadius(double radius)
{
   if (radius > 0)
      this->radius = radius;
}
```

Initial:

```
double Circle::getArea() const
{
   return M_PI * radius * radius;
}
```

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 ‘O’ 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 'O'. If a column has all 'O', what is the number of 'O'?

Define NUM_COLUMNS as a const with value 3. (Will not be graded for this step).

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

bool column_all_O(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] == 'O')
              num++;

       if (num == numRows)
          //condition (num == numRows) cannot be replaced by
          //(num == 3), the code work needs to for any 2d array
          return true;
   }

   return false;
```

}

Function main is optional. I attach for completeness.

```
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_O(arr, numRows) << endl;
    return 0;
}
```

Initial:

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)