Exam 2

Graded

Student

Zhangrui Weng

Total Points

73 / 100 pts

Question 1

Q1 **23** / 50 pts

1.1 **Q1.1** **5** / 5 pts

✔ **+ 5 pts** Correct

+ 0 pts Incorrect

1.2 **Q1.2** **0** / 5 pts

+ 5 pts Correct

✔ **+ 0 pts** Incorrect

1.3 **Q1.3** **5** / 5 pts

✔ **+ 5 pts** Correct

+ 0 pts Incorrect

1.4 **Q1.4** **0** / 5 pts

+ 5 pts Correct

✔ **+ 0 pts** Incorrect

1.5 **Q1.5** **0** / 5 pts

+ 5 pts Correct

✔ **+ 0 pts** Incorrect

1.6 **Q1.6** **0** / 5 pts

+ 5 pts Correct

✔ **+ 0 pts** Incorrect

1.7 **Q1.7** **0** / 1 pt

+ 1 pt Correct

✔ **+ 0 pts** Incorrect

1.8 **Q1.7 MEMORY** **1** / 1 pt

✔ **+ 1 pt** Correct

+ 0 pts Incorrect

1.9 **Q1.7 S/H 1** **1** / 1 pt

✔ **+ 1 pt** Correct

+ 0 pts Incorrect

1.10 **Q1.7 S/H 2** **1** / 1 pt

✔ **+ 1 pt** Correct

+ 0 pts Incorrect

1.11 **Q1.8** **0** / 1 pt

+ 1 pt Correct

✔ **+ 0 pts** Incorrect

1.12 **Q1.8 MEMORY** **1** / 1 pt

✔ **+ 1 pt** Correct

+ 0 pts Incorrect

1.13 **Q1.8 S/H 1** **0** / 1 pt

+ 1 pt Correct

✔ **+ 0 pts** Incorrect

1.14 **Q1.8 S/H 2** **1** / 1 pt

✔ **+ 1 pt** Correct

+ 0 pts Incorrect

1.15 **Q1.8 S/H 3** **1** / 1 pt

✔ **+ 1 pt** Correct

+ 0 pts Incorrect

1.16 **Q1.8 S/H 4** **1** / 1 pt

✔ **+ 1 pt** Correct

+ 0 pts Incorrect

1.17 **Q1.9** **0** / 1 pt

+ 1 pt Correct

✔ **+ 0 pts** Incorrect

1.18 **Q1.9 MEMORY** **0** / 1 pt

+ 1 pt Correct

✔ **+ 0 pts** Incorrect

1.19 **Q1.9 S/H 1** **0** / 1 pt

+ 1 pt Correct

✔ **+ 0 pts** Incorrect

1.20 **Q1.9 S/H 2** **0** / 1 pt

+ 1 pt Correct

✔ **+ 0 pts** Incorrect

1.21 **Q1.9 S/H 3** **1** / 1 pt

✔ **+ 1 pt** Correct

+ 0 pts Incorrect

1.22 **Q1.10** **1** / 1 pt

✔ **+ 1 pt** Correct

+ 0 pts Incorrect

1.23 **Q1.10 MEMORY** **1** / 1 pt

✔ **+ 1 pt** Correct

+ 0 pts Incorrect

1.24 **Q1.10 S/H 1** **1** / 1 pt

✔ **+ 1 pt** Correct

+ 0 pts Incorrect

1.25 **Q1.10 S/H 2** **1** / 1 pt

✔ **+ 1 pt** Correct

+ 0 pts Incorrect

1.26 **Q1.10 S/H 3** **1** / 1 pt

✔ **+ 1 pt** Correct

+ 0 pts Incorrect

Question 2

Q2

50 / 50 pts

Function Signature & Parameters

- ✔ **+ 2 pts** Function signature is unchanged.
- ✔ **+ 3 pts** Used values from parameter values. e.g. Did not use standard in (i.e. cin) to get any values.

+ 1 pt **Partial Credit**: Function signature is correct with one error.

+ 0 pts Hard coded answers from test cases.

+ 0 pts Got values from source other than the parameters.

Resize Array - Build & Copy

- ✔ **+ 3 pts** Uses row major ordering. Other rubric items give full credit if using column major instead of row major as long as it is consistent.
- ✔ **+ 3 pts** Correctly allocate memory for an array of pointers to arrays of int.
- ✔ **+ 0.5 pts** Creates the correct number of elements in array of pointers. (2*numRows) (all or none)
- ✔ **+ 3 pts** Correctly allocate memory for an array of int for each pointer.
- ✔ **+ 0.5 pts** Creates the correct number of elements in array of ints. (2*numCols) (all or none)
- ✔ **+ 3 pts** Each value from the original array is copied into the new array
- ✔ **+ 3 pts** Values are copied to the correct places (i.e. arr[i] is copied to newArr[2*i], newArr[2*i+1], newArr[2*i] and newArr[2*i+1]

+ 2 pts **Partial**: Correctly allocate array of pointers in some cases but not in others or there is a problem with the allocation in some cases.

+ 2 pts **Partial**: Correctly allocate array of ints in some cases but not in others or there is a problem with the allocation in some cases.

+ 2 pts **Partial**: Attempt to transfer data from old array to new array but there is a problem.

+ 1 pt **Partial**: Attempt to allocate array of pointers but there is a problem.

+ 1 pt **Partial**: Attempt to allocate array of int but there is a problem.

+ 1 pt **Partial**: Attempt to copy data from old array to new array but there are multiple problems.

Resize Array - Clean up

- ✔ **+ 3 pts** Correctly deallocate an array of ints (Can be on the wrong array)
- ✔ **+ 3 pts** Correctly deallocate an array of int*s. (Can be on the wrong array)
- ✔ **+ 2 pts** Deallocate arrays in sequence that won't generate a memory leak. i.e. deallocate dealocate the arrays of ints before the array of int*s. (Can be on the wrong array)

✔ **+ 2 pts** Deallocate the correct array. Note sometimes students end up deallocating the new array instead of the old one. (all or none)

+ 2 pts **Partial**: Attempt to deallocate array of ints but fail to indicate delete is for an array with [].

+ 2 pts **Partial**: Attempt to deallocate array of pointers but fail to indicate delete is for an array with [].

+ 1 pt **Partial**: Attempt to deallocate array of ints but there is a major problem.

+ 1 pt **Partial**: Attempt to deallocate array of pointers but there is a major problem.

+ 0 pts no evidence

Resize Array- Update Parameters

✔ **+ 2 pts** Update numRows and numCols

✔ **+ 1 pt** Updates arr to point to the new array

✔ **+ 1 pt** Updates arr at the right time (after (attempting to) delete arr)

+ 1 pt **Partial**: Attempt to update numRows and numCols, but at least one of them is set to the wrong value

+ 0.5 pts Partial: Updates arr incorrectly

Memory Errors

✔ **+ 2 pts** Avoid accessing outside array bounds. (All or none)

✔ **+ 3 pts** Avoid memory leaks (All or none)

Edge Cases

✔ **+ 3 pts** Throws something when numRows is 0, numCols is 0 or arr is nullptr

✔ **+ 2 pts** Throws an invalid_argument exception

+ 2 pts **Partial Credit** Throws something in some cases but not others

+ 0 pts Click here to replace this description.

Overall Correctness

✔ **+ 5 pts** The overall algorithm works to give a correct result excluding items already accounted for in prior rubric items.

+ 4 pts **Partial Credit**: The overall algorithm works but a minor logic error can result in an incorrect result.

+ 2.5 pts **Partial Credit**: Overall algorithm structure is in the right direction but details are unclear or does not work.

+ 1 pt **Partial Credit**: Major problems with the algorithm, but there are some elements that could be part of a correct solution.

+ 0 pts Not at all in the right direction

Partial Credit: Attempted and less than 5 points earned.

+ 0 pts No additional partial credit needed.

+ 5 pts **Partial Credit**: Attempted and no other points earned.

+ 4.5 pts **Partial Credit**: Attempted and 0.5 points earned.

+ 4 pts **Partial Credit**: Attempted and 1 point earned.

+ 3.5 pts **Partial Credit**: Attempted and 1.5 points earned.

+ 3 pts **Partial Credit**: Attempted and 2 points earned.

+ 2.5 pts **Partial Credit**: Attempted and 2.5 points earned.

+ 2 pts **Partial Credit**: Attempted and 3 points earned.

+ 1.5 pts **Partial Credit**: Attempted and 3.5 points earned.

+ 1 pt **Partial Credit**: Attempted and 4 points earned.

+ 0.5 pts **Partial Credit**: Attempted and 4.5 points earned.

Exam 2

Version: R0530

Academic Integrity

- Aggies do not lie, cheat, or steal, nor tolerate those who do.
- We hope you have fun solving these problems, even though this is an exam ☺
- You can do this!

Exam Guidelines

- You will have 50 minutes to complete and submit this exam.
- Do not write outside of the margin box. We scan the exams and anything outside those lines will likely get cut off.
- You cannot use any electronic devices (including calculators, phones, smart watches, and computers)
- You may use
    - A writing utensil (e.g. pen/pencil)
    - Scratch Paper that we provide.
        - § Scratch paper may not have the margin box, so leave a margin around the edge of your scratch paper to avoid having information cut off when scanned.
    - Up to 5 pages of exam aids.
        - § Exam aids can be pages up to 8.5X11 inches and can be handwritten or printed on both sides.
        - § Do not use exam aids for scratch work.
            - If you use an exam aid for scratch work, submit with exam.
- Start Exam when prompted.
- Stop exam when you are finished or when time expires.
    - Attach any scratch work to the end of your exam. Including exam aids used for scratch work.
    - Submit your exam.
        - § You may keep your exam aids if they were not used for scratch work.

Fill in name and UIN before exam starts

Name: Zhaoyu Wang

UIN: 832009830

Name:_____

Q1 - Code Tracing (50 points)

Overview

- In the following parts you will be given small pieces of code. Determine what each piece of code prints, and fill in the blank.
- For each piece of code, assume that the file has `#include <iostream>` and `using std;:cout;`

Q1.1

```
int alpha(int x, int& y){
      x++;
      y = y*2;
      return x + y;
}
int main(){
      int a = 1;
      int b = 2;
      int c = alpha(a,b);
      cout << a << b << c;
}
```

alpha

Var			
int x	1+1	2	
int& y	2	4	return 6

main

Var	
int a	1
int b	2
int c	6

This code prints: 146

Q1.2

```
void beta(int& x, int* y){
      *y = 5;
      y = &x;
      (*y)++;
}
int main(){
      int a = 1;
      int b = 6;
      int* c = &b;
      beta(a,c);
      cout << a << b;
}
```

beta

main

int a	1
int b	3
int *c	

This code prints: 13

Name:_____

Q1.3

```
void delta(int* x){
      x[2] = 5;
      x = &(x[2]);
}
int main(){
      int arr[] = {2,4,6,8,10};
      delta(arr);
      cout << arr[1] << arr[2];
}
```

This code prints: 45

Q1.4

```
void gamma(int& x, int*& y){
      x = *y;
      x++;
      *y = 5;
      y = &x;
}
int main(){
      int a = 1;
      int* b = new int(3);
      gamma(a,b);
      cout << a << *b;
}
```

This code prints: 43

Name:_____

Q1.5

```
void zeta(double*& x){
      x[2] = 5;
      x = new double[7]{};
}
int main(){
      double* arr = new double[3]{1.62,2.72,3.14};
      zeta(arr);
      cout << arr[2];
}
```

This code prints: _____

Q1.6

```
void epsilon(char* x){
      for(int i = 0; x[i] != '\0'; i++){
            x[i] = x[i+1];
      }
      x = x+1;
}
int main(){
      char* str = new char[10]{"Howdy!"};
      epsilon(str);
      cout << str;
}
```

This code prints: _____

Name:_____

Q1.7

```
int main(){
      int x = 3;
      if(x > 2){
            int* y = &x;
            *y = 7;
      }
      cout << x;
}
```

This code prints: 3

Does this code have a memory leak? (write "yes" or "no") No

Fill the table to identify where the variable is stored, i.e. fill the table with either Stack or Heap:

	x	y
Stack/Heap	Stack	Stack

Q1.8

```
int main(){
      char* x = new char[3]{'a','b','c'};
      char y[3]{'d','e','f'};
      char* z = y;
      z[1] = 'g';
      cout << y[1];
      z = x;
      delete[] z;
      x = nullptr;
      z = nullptr;
}
```

This code prints: e

Does this code have a memory leak? (write "yes" or "no") No

Fill the table to identify where the variable is stored, i.e. fill the table with either Stack or Heap:

	x	z	x[1]	z[1]
Stack/Heap	Heap	stack	Heap	Stack

Name:_____

Q1.9

```
int main(){
      long** x = new long*[7];
      for(int i = 0; i < 7; i++){
            x[i] = new long(i*i);
      }
      cout << x[2][0];
      for(int i = 0; i < 7; i++){
            delete x[i];
      }
}
```

0 1 2 3 4 6

This code prints: 2

Does this code have a memory leak? (write "yes" or "no") No

Fill the table to identify where the variable is stored, i.e. fill the table with either Stack or Heap:

	x	x[2]	x[2][0]
Stack/Heap	Heap	Stack	Heap

Q1.10

```
int main(){
      int** x = new int*;
      *x = new int(7);
      int** y = x;
      **y = 5;
      cout << **x;
      delete x;
}
```

This code prints: 5

Does this code have a memory leak? (write "yes" or "no") Yes

Fill the table to identify where the variable is stored, i.e. fill the table with either Stack or Heap:

	x	*x	**x
Stack/Heap	Stack	Heap	Heap

Name:_____ 7 / 8

Q2 - Dynamic Array Scaling (50 points)

Overview

Given a dynamic array of integers with n rows and m columns, rescale it to have 2n rows and 2m columns by replacing each value k in the array with a 2 x 2 square of entries, all of which have the same value k (This operation is useful, for example, when resizing an image).

Requirements

Write the function `void scaleArray(int**& arr, unsigned int& numRows, unsigned int& numCols)` which replaces each entry of the array with a 2x2 square of the same entry, causing the array to double in both dimensions.

- Use row-major ordering when working with your 2D array.
- You must resize the array.
- You cannot have any memory errors.
- Throw an instance of std::invalid_argument (with whatever error message you want) if any of the following is true:
  - arr is nullptr
  - numRows is 0
  - numCols is 0
- You may not use any libraries besides stdexcept

Examples

- If `scaleArray(arr, numRows, numCols)` is called with
  - arr
    -
      ```
      12 20 33
      41 52 67
      ```
  - numRows is 2
  - numCols is 3
- After calling the function
  - arr
    -
      ```
      12 12 20 20 33 33
      12 12 20 20 33 33
      41 41 52 52 67 67
      41 41 52 52 67 67
      ```
  - numRows is 4
  - numCols is 6

Name:_____ 8 / 8

Q2 Answer (Dynamic Array Scaling)

```
#include <stdexcept>
void scaleArray(int**& arr, unsigned int& numRows, unsigned int&
numCols)
if(arr == nullptr || nrows <= 0 || ncolumns <= 0)
{ throw std::invalid_argument("Invalid argument")
}
// create new array
int** temp = nullptr;
int rows = 2 * numRows, columns = 2 * numCols;
temp = new int*[rows];
for(int i = 0; i < rows; i++)
{ temp[i] = new int[columns]{0};
}
// algorithm
// (2i, 2j) (2i, 2j+1) (2i, 2j+2)
// (2i+1, 2j) (2i+1, 2i+1) (2i+1, 2j+2)
// (2i+2, 2)
for (int i = 0; i < nrows; i++) {
for (int j = 0; j < ncolumns; j++) {
    int x = arr[i][j];
    temp[2*i][2*j] = x;
    temp[2*i][2*j+1] = x;
    temp[2*i+1][j*2] = x;
    temp[2*i+1][j*2+1] = x;
}
```

Next Page for More code

```
// release the memory from old array
for(int i=0; i<nrows; i++){
    delete[] arr[i];
}

delete[] arr;

arr = temp;
numRows = row;
numColums = colums;
}
```