## Problem Set 2

As in the previous problem set, here is a program for your consideration. Please answer the questions that follow by looking only at the source code.

```
#include
using namespace std;

int main() {
 int Boys = 3, Girls = 5;
 void F1(int males, int females);
 void F2(int &m, int &f);

 F1(Boys, Girls);
 cout << "\nAfter calling F1, within main()";
 cout << "\n\tBoys = " << Boys; // #2
 cout << "\n\tGirls = " << Girls;

 F2(Boys, Girls);
 cout << "\nAfter calling F2, within main()";
 cout << "\n\tBoys = " << Boys; // #4
 cout << "\n\tGirls = " << Girls;
}

void F1(int b, int g) {
 b += 3, g += 4;
 cout << "\nF1";
 cout << "\n\tBoys = " << b; // #1
 cout << "\n\tGirls = " << g;
}

void F2(int &b, int &g) {
 b = b + 8, g = g + 5;
 cout << "\nF2";
 cout << "\n\tBoys = " << b; // #3
 cout << "\n\tGirls = " << g;
}
```

Question 1: What is the output of the Boys variable on the marked lines?

| | |
|---|---|
| A) | #1: 6<br>#2: 3<br>#3: 11<br>#4: 11 |
| B) | #1: 6<br>#2: 3<br>#3: 11<br>#4: 3 |
| C) | #1: 6<br>#2: 6<br>#3: 11<br>#4: 11 |

| | |
|---|---|
| | #4. 11 |
| D) | It outputs nothing because it does not compile or run. |

Question 2: Choose all that apply relating to the following lines of the program:

```
void F1(int males, int females);
void F2(int &m, int &f);
```

| | |
|---|---|
| A) | C++ rules state that we can remove these two lines as long as the methods are defined before use. |
| B) | C++ rules state that the argument names must be the same between the declaration and definition. |
| C) | This program will crash if we remove these two lines. |
| D) | It's more common for the declarations to be stated in the global scope. |
| E) | These are called forward declarations. |

Question 3: If we move the following line from main() and place it in the global scope, what will happen?

```
int Boys = 3, Girls = 5;
```

| | |
|---|---|
| A) | The output would be the same. |
| B) | Boys would = 3 and Girls would = 5 in all output |
| C) | Boys would = 3 and Girls would = 5 only in the output from main() |

Question 4: What if we changed the beginning of the program to look like this:

```
// We have moved moved these to global scope
const int Boys = 3;
const int Girls = 5;

void main() {
  //int Boys = 3, Girls = 5;
```

| | |
|---|---|
| A) | The program would compile but will crash when we try and run it. |
| B) | There would be no changes in the output |
| C) | The output would be Boys = 3 Girls = 5 throughout the program |
| D) | The output would be Boys = 3 Girls = 5 only in the output from main() |
| E) | The program will probably not compile (depending on the compiler). |

Question 5: Data is passed by value in F2.

| | |
|---|---|
| A) | True. |
| B) | False. |

## Problem Set 3

As in the previous problem set, here is a program for your consideration. Please answer the questions that follow by looking only at the source code. This one is more interesting than the previous two - trace the code carefully.

```
  #include
using namespace std;

const int MAX_SIZE = 20;
typedef int ARR2D[MAX_SIZE][MAX_SIZE];

void Print(ARR2D in_array, int rows, int cols);
void Fill(ARR2D in_array, int rows, int cols);

int main() {
  ARR2D matrix;
  int row, col;
  do {
    cout << "Please enter the size of the matrix to generate (rows and cols) :" << endl;
    cin >> row >> col;
  } while (row <= 0 || row > MAX_SIZE || col <= 0 || col > MAX_SIZE);
  Fill(matrix, row, col);
  Print(matrix, row, col);
  return(0);
}

void Print(ARR2D in_array, int rows, int cols) {
  for (int i = 0; i < rows; i++) {
    for (int j = 0; j < cols; j++)
      cout << '\t' << in_array[i][j];
    cout << endl;
  }
}

void Fill(ARR2D in_array, int rows, int cols) {

  for(int i = 0; i < rows; i++)
    for (int j = 0; j < cols; j++)
      in_array[i][j] = 0;

  const int limit = rows * cols;
  int cNum = 1;
  int cRow = 0;
  int cCol = 0;
  int cDir = 0;  // 0-north, 1-east, 2-south, 3-west

  while(true) {
    // Place the number.
```

```
    in_array[cRow][cCol] = cNum;
    cNum++;
    if (cNum > limit) break;

    int fRow = cRow;
    int fCol = cCol;
    while (true) {
      fRow = cRow;
      fCol = cCol;
      switch(cDir) {
        case 0: fRow--; break;
        case 1: fCol++; break;
        case 2: fRow++; break;
        case 3: fCol--; break;
      }

      if ( fRow >= 0 && fRow < rows && fCol >= 0 && fCol < cols && in_array[fRow][fCol] == 0)
        break;
      cDir = (cDir + 1) % 4;
    }
    cRow = fRow;
    cCol = fCol;
  }
}
```

Question 1: What does this program output with input of 3 for rows and 4 for columns?

| | |
|---|---|
| A) | 1 2 3<br>4 5 6<br>7 8 9<br>10 11 12 |
| B) | 1 2 3 4<br>5 6 7 8<br>9 10 11 12 |
| C) | 12 11 10 9<br>8 7 6 5<br>4 3 2 1 |
| D) | 1 3 2 4<br>8 6 7 5<br>9 11 10 12 |
| E) | 1 2 3 4<br>10 11 12 5<br>9 8 7 6 |
| G) | 9 8 7 6<br>10 11 12 5<br>1 2 3 4 |
| H) | It does not output anything - the logic is faulty. |
| I) | It does not output anything - there are syntax errors. |
| | |

| | |
|---|---|
| J) | It does not output anything - it's not supposed to. |
| K) | It outputs the first 12 numbers that come to mind as you are waiting for the program to run. |

Question 2: What if we added the following line to our main() function?

```
MAX_SIZE = 10;
```

| | |
|---|---|
| A) | This is not allowed in C++. |
| B) | This is allowed; the program would run with MAX_SIZE set to 20 |
| C) | This is allowed; the program would run with MAX_SIZE set to 10. |

Question 3: Consider the following four lines from the program above:

```
const int MAX_SIZE = 20;
typedef int ARR2D [MAX_SIZE][MAX_SIZE];

void Print  (ARR2D A, int rows, int cols);
void Fill   (ARR2D A, int rows, int cols);
```

1) Is it possible to use a const in a typedef?
2) Is it possible to use a typedef in a declaration before a variable has been declared of that type?

| | |
|---|---|
| A) | 1) Yes 2) Yes |
| B) | 1) No 2) No |
| C) | 1) No 2) Yes |
| D) | 1) Yes 2) No |

Question 4: Can we use the following:

```
#define MAX_SIZE 20
```

instead of:

```
const int MAX_SIZE = 20;
```

| | |
|---|---|
| A) | Yes it will work and it's fine to use #define for constants in C++ |
| B) | Yes it will work but we usually do not to use #define for constants in C++ |
| C) | #define is not available in C++ |
| D) | You can't do either of these things in C |

Question 5: typedef is used to create an alias for a type name.

| | |
|---|---|
| A) | True. |
| B) | False. |

Question 6: What would happen if we did not initialize the array to 0 in the Fill() function?

| | |
|---|---|
| A) | It will run but the output will be all 12's |
| B) | It will run fine and generate the same output as if the array were initialized to 0 |
| C) | The program will not run or will crash |
| D) | It will run but the output will be all 0's |
| E) | It will run but may not generate any output |

Question 7: Check all that apply. Why do we use const for MAX_SIZE in this program? Isn't it easier to just type "20" instead of MAX_SIZE wherever it is needed?

| | |
|---|---|
| A) | MAX_SIZE is a built-in C++ value that anyone can use. Just set it and use it. |
| B) | Global consts should be avoided just like global variables |
| C) | Using a const makes our program easier to understand |
| D) | Magic numbers in a program are usually considered good practice. |
| E) | If we want to change MAX_SIZE, we only need to change it in one place |

Question 8: The switch statement in the Fill() function should have a default case, because it's considered good style to include one.

| | |
|---|---|
| A) | True. |
| B) | False. |

Question 9: Notice in the Fill() function, we declare variables in between statements. For example, cNum and cRow are declared and initialized after a for-loop has run. Will this work in C++, or do all the variables have to be declared at the top of the function?

| | |
|---|---|
| A) | It's fine to do this. |
| B) | All the variables must be declared at the top of the function. |
| C) | Both ways are wrong - C++ doesn't allow variables anywhere in a program. |
| D) | All variables must be declared in the global scope. |