COP 3014 - Programming 1 - Fall 2021

Sample Questions and Instructions for the Third Milestone Assignment

Milestone Assignment 3 on Thursday 12/09/2021 at 3:00 PM, in class

The Milestone Assignment consists of

1. 10 multiple choice questions - 30 points
2. 1 "find the output" question - 15 points
3. 2 code writing question - 25 + 15 = 40 points
4. 1 code debugging question - 15 points
5. 4 short answer questions - 20 points

General details:

- **The Milestone Assignment will be on paper, in-class**
- The milestone assignment can be solved in about 60 minutes. You will have 120 minutes as long as you're in class at 3:00 PM.
- The milestone assignment is supposed to be individual work. You will be asked to sign an Honor Code statement.
- The Milestone Assignment is closed-book, closed-notes. You are **NOT** allowed to use any electronic devices, including graphing calculators.
- You will have an opportunity to earn 20 extra credit points.
- The milestone assignment will examine your understanding of the material rather than your capacity to memorize copious amounts of information.
- **The test is completely cumulative.**
- Please try and attempt all questions. You get points for trying.
- Anything from the homeworks / quizzes / in class examples / exercises / slides is fair game. You don't need to look for more material.
- Code debugging is mostly syntax based (typos, missing brackets, semicolons, mismatched quotes, etc.)
- The code writing questions will be heavily based on the homeworks and class examples and exercises, with some modifications.
- The multiple choice and the debugging questions will test your familiarity with the C++ language and syntax. The code writing questions will test your knowledge of programming.
- **For Problem 2, you would need to explain you answer. Just writing out your output will only get you partial credit.**
- **For Problem 3, you will be graded based on**

  - Whether your code works
  - Are you using the required concepts?
  - Are you adhering to standard programming practices discussed in the course (no goto, auto, break outside a switch, continue; declaring and defining functions, etc).
  - IT IS MORE THAN JUST AN OUTPUT MATCHING PROBLEM
- **For Problem 4, you need to fix the error and explain what caused the error.** You will only be tested on syntax and semantic errors.
- **Solutions to the short answer questions must be SHORT and to the point. A 6 paragraph essay when 4 lines would do means your answer will just be skimmed through in the interest of grading time (The needs of the many ...)**
- You will also receive a Canvas announcement about the milestone assignment with these details on Sunday, 12/05/2021.
- **General Advice: For the sake of completing the assignment on time, please answer all questions before seeking to verify the answers**.
- Making me laugh might gain you points (depends on the quality of the joke).

Topics to study

- Basic C++ Syntax
  - Writing a basic C++ program stub: including required libraries, adding namespaces and writing the main function.
  - Simple statements - syntax.
  - Comments.
  - Reserved words, literals and escape sequences.
  - Style guidelines.
- Primitive Data types, variables, operators, and sequential execution.
  - Naming, declaring and initializing variables.
  - Primitive data types.
  - Type Conversions - implicit and explicit.
  - Arithmetic Operators and operator precedence.
- I/O - printing and reading values from the user.
  - cout statements - printing literals and variables.
  - Precision for floating point variables.
  - Using cin to read data of different kinds.
- Selection statements and loops
  - Relational and logical operators.
  - Writing simple, multiple and nested if statements.
  - switch - case statements.
  - while, do-while and for loops
  - break and continue statements.

- Writing functions in C++.
  - Writing simple functions.
  - Passing arguments and returning values. Pass by value and pass by reference.
  - Scope of local and global variables.
  - Function overloading and default parameters.
- Arrays
  - Declaring and initializing an array.
  - Reading in array values from the user and printing arrays.
  - Basic array operations - looking for a number, math with array elements, etc.
  - Passing arrays as parameters to functions.
  - Multi dimensional arrays.
- Strings
  - Cstrings - arrays of characters.
  - string objects
  - Reading strings with getline - both options
  - cstring, cctype and string libraries
  - Basic string operations - counting different kinds of characters, finding and replacing substrings, etc.
  - Declaring, initializing, comparing and concatenating strings
  - Arrays of strings.
- Pointers
  - Declaring, initializing and dereferencing pointers.
  - null pointer, reinterpret cast and pointer arithmetic.
  - Pass by address, arrays with pointers, cstrings with pointers.
  - new and delete operators, dynamic memory allocation.
- Structures
  - Creating structures, declaring variables of the new type, creating nested structures, arrays of structures, pointers to structures. Creating dynamic structures.
  - The dot and assignment operators.
  - Structures and functions - passing and returning.
- File Operations
  - Text Files and Streams
  - How to read from a text file, how to write to a text file.
  - Objects of ifstream and ofstream and some associated member functions, like open(), eof() and close().
  - Character I/O - ignore()
  - Streams and functions
- You don't need to study from outside sources. The test is made entirely from the notes, quizzes and assignments.

Sample Questions

1. Which of the following is a C++ reserved word?

   (a) `file`

   (b) `ifstream`

   (c) `ofstream`

   (d) `struct`

   Answer: `struct`, the rest are identifiers.

2. Which of the following strings will be placed before "banana" in lexicographical order?

   (a) Peach

   (b) Strawberry

   (c) 2 green apples

   (d) All of the above

   Answer: All of the above (Numbers first, then uppercase letters, then lowercase)

3. Consider the following function declaration:
   `int count (char *st)`
   If you had a for loop (loop variable i) iterating through the C-String st in the function, which of the following lines would result in an error?

   (a) `cout<<st[i];`

   (b) `st[i++] = 'X';`

   (c) `cout<<*st[i];`

   (d) `cout<<*(++st);`

   Answer: `cout<<*st[i];` - This is a double dereference on a single pointer.

4. A perfect number is defined as a positive integer that is equal to the sum of its factors (not including itself). For example, 6 is a perfect number, because the sum of the factors of $6 = 1 + 2 + 3 = 6$. Write a C++ program to print all the perfect numbers between 2 and 1000 (inclusive)
   Sample Run:

   ```
   6
   28
   496
   ```

   Solution:

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

   int main()
   {
       for(int num=2; num<=1000; num++)    //go through numbers one at a time
       {
           int sumFactors=0;
           for(int i=1; i<num;i++)    //go through all numbers <= num
           {
   ```

```
            if(num%i == 0)  //found a factor
                sumFactor += i;     //add it to sum variable
        }
        if(num == sumFactor)
            cout<<num<<endl;
    }
    return 0;
}
```

5. Write a C++ program to create an array of 200 integers. Read in the values from a file called `input.txt`. Then, print all the numbers that occur more than once. You can assume the file will open without issues. You can assume the file has 200 numbers. It is ok if the number is printed as many times as it occurs.
   Solution:

```
#include<fstream>
%using namespace std;
int main()
{
    ifstream in;
    int array[200];

    in.open("input.txt");
    if(!in)
    {
        cout<<"Issues opening file input.txt"<<endl;
        return 0;
    }
    //file is good
    for(int i=0; i<200; i++)
        in>>array[i];

    in.close(); //close the file

    for(int i=0; i<200; i++) //go through the whole array
    {
        for(int j=0; j<200; j++) // check it against every other element that is not itself
            if(i!=j && array[i] == array[j])
                cout<<array[j]<<endl;
    }
    return 0;
}
```

6. Write a C++ program with the following requirements:
   - Create a structure called `Date` that contains the following elements:
     - day - integer
     - month - integer
     - year - integer
   - Create a structure called `Flight` that contains the following elements:
     - Flight Number - string
     - Source - string
     - destination - string

  - date - Date
  - number of passengers - integer
- In the main function, create a structure variable and read in values from the user.
- Write a function to accept a structure variable and prints the contents of a the structure in sentence form.

Sample Run:

```
Enter the following details:
Flight Number: DL 110
Source: ATL
Destination: LAX
Date (month day year): 12 12 2017
Number of passengers: 120

DL 110 from ATL to LAX on 12/12/2017 has 120 passengers.
```

Solution:

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

struct Date{
    int day,month,year;
};
struct Flight{
    string flNo,source,dest;
    int numPas;
    Date date;
};

void print(Flight f);

int main()
{
    Flight fl;
    cout<<"Enter the following details:"<<endl;
    cout<<"Flight Number: ";
    getline(cin,fl.flNo);
    cout<<"Source: ";
    getline(cin,fl.source);
    cout<<"Destination: ";
    getline(cin,fl.dest);
    cout<<"Date (month day year): ";
    char slash;
    cin>>fl.date.month>>slash>>fl.date.day>>slash>>fl.date.year;
    cout<<"Number of passengers: ";
    cin>>fl.numPas;

    print(fl);
    return 0;
```

```
}

void print(Flight f)
{
    cout<<f.flNo<<" from "<<f.source<<" to "<<f.dest<<" on "
        <<f.date.month<<"/"<<f.date.day<<"/"<<f.date.year<<" has "
        <<f.numPas<<" passengers."<<endl;
}
```

7. What is a reference variable? Explain with an example.
   Solution: A formal parameter that is just a nickname for the actual parameter. Any changes made to the formal parameter will be made to the actual parameter as well.
   Eg: void change(int ref);

8. Find the output of the following code snippet:

```
char st[] = "Once upon a time there was an ugly barnacle";
for(int i=0; i<strlen(st); i++)
        st[i++]=toupper(st[i]);
        cout<<st<<endl;
```

   Solution: OnCe uPoN A TiMe tHeRe wAs aN UgLy bArNaClE
   The loop iterates through every character in the string. It grabs a character, gets the uppercase form of that character, and put's it in the same spot. Then the post-increment moves it to the next character. Then, the loop increment moves it over again. This results in only alternate characters being turned into uppercase.

1 How to actually take the Milestone Assignment

This is a suggested method for tackling the milestone assignment and the key to completing it on time. Please note the emphasis on testing of your understanding of the material - programming is a process, so we're testing your understanding of the processes involved to accomplish the task using the tools you've learned, not your memorization skills.

1.1 Before the assignment

Practice is the only way to get better at programming. Please solve the practice problems. And then try to solve variations of the practice problems ON PAPER. For example, if the practice problem asked you to find the largest number, also try and find the smallest. If the practice problem asks you to run a loop forwards, also try running it backwards.

Similarly work through the class examples and homeworks. The problems on the test are heavily based on class examples, homework problems or practice problems.

If you have questions, ask during help sessions or after class. Help Sessions are not just for homework help. They're a vital resource when you don't have regular in-person contact with your instructor or TA/LA.

1.2 During the assignment

TIME YOURSELF

Also, scratch paper and pencils are your best friends.

- 1 minute per multiple choice question - you either know it or can deduce it within a minute. Otherwise, just put down an answer and mark the question as a "review later" on your scratch paper.
- 10 minutes for the code reading question - work out the process on scratch paper. The "explanation" part can be short - explain the problem solved by the program - a line by line explanation is not required.
- 20 minutes for each of the code writing problems. Turn in what you have. There is a lot of partial credit.

  For this problem, you don't need to do error checking. You only need to solve the specific problem, assuming everything else goes well.
- 10 minutes for the code debugging question - we're only looking for syntax errors - errors that would stop CLion from compiling your program. Do not try to find the output for the program, or even see if the program makes sense line by line. Syntax errors are usually restricted to that line.

  You need to tell me what the error is and how it can be fixed - or show a line with the error fixed.

  We're aware errors propagate. So, if a future line will have an error due to previous line, and you've already identified and fixed the first, then this one wouldn't count. For example, if there is an error in a line with a variable declaration, then fixing that line would fix all the future "uninitialized variable" errors.
- 5 minutes per short answer question - Here, we're interested in what you know about the concept.
- This would leave you with 30 minutes to finish up any unfinished problem, and review.
- The objective of the test is not to be 100% right. It mimics the programming interview, where a "instinctual" solution is good enough for the first part of the process - your initial thoughts - that demonstrates your level of knowledge and understanding of programming.