CS 101 Autumn 2023 — Midsem — 22 Sep 2023

10 questions, 100 marks (Instructor: Suyash P. Awate)

Roll Number	ANSWERS
Division and Group	
Name	

Q. No.	Marks	Graded By	Verified By	Student's Cribs
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
TOTAL				

Please read the following instructions carefully before you start.

- Write your roll number, name, and group number on this page in the space provided. A paper without a roll number and name will NOT be graded.
- Write your answers neatly with a blue/black pen on this question paper itself in the space provided for each question. At the end, you must submit this paper to the invigilator.
- Rough pages will NOT be provided. Use the empty space in the margins.
- Please note that your answers should NOT include any programming concept that hasn't been covered in the class so far. If such answers are found, they shall NOT be graded.
- No clarifications will be provided on any question. When in doubt, make suitable assumptions, state them clearly, and proceed to solve the problem.
- All the best!

1. [7 points] The following code snippet is intended to display all the prime factors of an integer larger than 1. For example, for input as “120”, the output is intended to be “prime factors of 120 are: 2 2 2 3 5”.

   However, some of the lines in the code given below contain an error. Your task is to: Identify incorrect statements and write down correct ones with their line numbers, in the space provided to the right.

   Guidelines: You cannot replace/remove the “while” statements. You cannot replace/remove the “cout” statements. You can only replace the expressions contained within the condition and/or the body. You cannot add any statement or remove any statement.

Code	Line No. and correct expression

`1. main_program {`
`2:   unsigned int n;`
`3:   cin >> n;`
`4:   cout << "prime factors of n are: ";`
`5:   unsigned int i = 1;`
`6:   while (i < n)`
`7:   {`
`8:      while (n % i = 0)`
`9:      {`
`10:        cout << 'i' << " ";`
`11:        n = n % i;`
`12:     }`
`13:     i = i + 2;`
`14:  }`
`15: }`
	One point for each correct expression
	Line 4:cout << "prime factors of " << n << " are: "
	
	Line 5: unsigned int i = 2;
	
	Line 6: while (i <= n)
	
	Line 8: while (n % i == 0)
	
	Line 10: cout << i << " ";
	
	Line 11: n = n / i
	
	Line 13: i = i + 1

2. [10 points] Consider the following code snippet:

```
main_program {
  for (int i = 0; i < 8; i++)
  {
    if (i%2 == 0) cout << i + 1 << " ";
    else if (i%3 == 0) { cout << i*i << " "; continue; }
    else if (i%5 == 0) { cout << 2*i -1 <<" "; break; }
    else cout << i << " ";
  }
  cout << "EOP";
}
```

   (a) What will be the output of the above program? [3 points]

   1 1 3 9 5 9 EOP

   (b) How many times will the for loop iterate? [1 point]

   6

   (c) For i=7, which control statement(s) will be executed? [1 point]

   No control statement will be executed.

(d) If you wanted to change the loop to count downwards from 10 to 1, what modifications would you make? [2 points]

```
for (int i = 10; i >= 1; i--)
OR
for (int i = 10; i > 0; i--)
```

(e) What will be the output of the program with the new modifications made in part (d) above? [3 point]

```
11 81 9 7 7 9 EOP
```

3. [8 points] For the program given below, answer the sub-parts, a, b, c, and d:

```
main_program {
    int x = 0;
    int s = 0;
    cin >> x;
    while(true) {
        if(s == 0 && x == 1) { s = 1; cin >> x; continue; }
        if(s == 1 && x == 0) { s = 2; cin >> x; continue; }
        if(s == 1 && x == 1) { break; }
        if(s == 2 && x == 0) { s = 1; cin >> x; continue; }
        if(s == 2 && x == 1) { s = 2; cin >> x; continue; }
    }
}
```

For each of the sub-parts, the program may have any of the following behaviors:

Statement 1: The program does not terminate and waits for the next input from the user

Statement 2: The program terminates without taking any user input

Statement 3: The program does not terminate and goes into an infinite loop

Statement 4: The program accepts all the given inputs, does some action on each of them, and then terminates.

Statement 5: The program terminates without taking some of the inputs

Write Statement numbers (1, 2, 3, 4, or 5) in the blanks provided: Note: The input is given to the program in a space-separated manner. 2 points each

(a) If the inputs to the above program are: 0 1 0 0 1, then Statement ___3___ is valid

(b) If the inputs to the above program are: 1 0 0 1, then Statement ___4___ is valid

(c) If the inputs to the above program are: 1 0 0 0 1 1 1 0, then Statement __1__ is valid

(d) If the inputs to the above program are: 1 0 1 0 1 1 1, then Statement __5__ is valid

4. [10 points] Write down the complete output of the following program:

```
main_program {
    int i, j, k;
    int lines = 5;
    int num = 1;
    char c = 'A';

    for (i=0; i<lines; i++) {
        for (j=0; j<lines-i; j++)
            cout<<"*";

        for (k=0; k<=(num/2); k++)
            cout<<c++;

        c = c - 2;

        for (k=0; k<(num/2); k++)
            cout<<c--;

        for (j=0; j<lines-i; j++)
            cout<<"*";

        num = num + 2;
        c = 'A';
        cout << endl;
    }
}
```
	2 points for each correct line
	```
	*****A*****
	****ABA****
	ABCBA
	ABCDCBA
	*ABCDEDCBA*
	```

5. [10 points]

An anagram of an n-digit number is defined as another n-digit number that consists of the same digits as the original but in a different order. For example, 1342 and 4321 are anagrams of the number 1234. The code below outputs whether the two 4-digit numbers `n1` and `n2` are anagrams of one another or not. Please go through the code and fill the blanks with the correct statements.

```
main_program {

    // we assume that n1 and n2 are 4-digit positive integers
    int n1, n2;
    cin >> n1 >> n2;

    int c1, c2, c3, c4, c5, c6, c7, c8, c9, c0;

    // 1 point
    c1 = c2 = c3 = c4 = c5 = c6 = c7 = c8 = c9 = c0 = _____0_____;
    for ( int d=0; d<4; d++ ) {  // 1 point

        switch (n1 % 10) {
            case 0: c0++; break;            case 1: c1++; break;
            case 2: c2++; break;            case 3: c3++; break;
            case 4: c4++; break;            case 5: c5++; break;
            case 6: c6++; break;            case 7: c7++; break;
            case 8: c8++; break;            case 9: c9++; break;
        }
        switch (n2 % 10) { // 3 points

            case 0: c0--; break;            case 1:  c1--; break;

            case 2: c2--; break;            case 3:  c3--; break;

            case 4: c4--; break;            case 5:  c5--; break;

            case 6: c6--; break;            case 7:  c7--; break;

            case 8: c8--; break;            case 9:  c9--; break;
        }

        n1 = n1/10; // 1.5 points

        n2 = n2/10;  // 1.5 points
    }

    // 2 points
    if ( (c0 == 0) && (c1 == 0) && (c2 == 0) && (c3 == 0)

         && (c4 == 0) && (c5 == 0) && (c6 == 0) && (c7 == 0)

         && (c8 == 0) && (c9 == 0) ) {
        cout << "numbers are anagrams";
    }
    else {
        cout << "numbers are not anagrams";
    }
}
```

6. [10 points] Determine the value of each of the indicated variables AFTER the following code snippet executes. Assume that each integer occupies 4 bytes and that `m` is stored in memory starting at byte `0x3fffd00`.

Code:	Write the value of each of these variables after the entire code is executed 1 point for each correct output
`int m = 44;`	(a) `m` = 46
`int* p = &m;`	(b) `n` = 44
`int& r = m;`	(c) `&m` = 0x3fffd00
`int n = (*p)++;`	(d) `*p` = 46
`int* q = p;`	(e) `r` = 46
`r = --(*p) + 1;`	(f) `*q` = 46
`++*q;`	(g) `&r` = 0x3fffd00
`int **x = &p;`	(h) `p` = 0x3fffd00
	(i) `q` = 0x3fffd00
	(j) `*x`= 0x3fffd00

7. [20 points] Consider the following code where `m` and `n` are greater than `0`:

```
int m, n; cin >> m >> n;
statement-set-1;
for (int i = 0; i < n; i++) {
    statement-set-2;
    for (int j = 0; j < m; j++) {
        statement-set-3;
    }
    statement-set-4;
}
statement-set-5;
```

Instructions: Modify the given code using equivalent code structures that use loops as described below. Ensure that

- The modified code maintains exactly the same scope for all existing variables (i.e., `m`, `n`, `i`, `j`) at ALL points within the code (i.e., for `statement-set-1`, `statement-set-2`, `...`, `statement-set-5`). For example, at:
  - `statement-set-1 and statement-set-5`, only `m` and `n` are accessible.
  - `statement-set-2 and statement-set-4`, the variables `m`, `n`, and `i` are accessible, while the variable `j` is not accessible.
  - `statement-set-3`, all variables i.e. `m`, `n`, `i`, and `j` are accessible.
- The modified code cannot create any additional variables in sub-parts (a), (b), (c), and (d). You may create/modify additional variables in sub-part (e).
- The modified code cannot contain any if statements or any conditional expressions or any function calls for sub-parts (a), (b), (c), and (d). In the sub-part (e), you may use the `if` statement and any counter variable such as `i` or `j`.

(a) [3.5 points] Replace the outer `for` loop with a `while` loop

```
int m, n; cin >> m >> n;
statement-set-1;
{
    int i = 0;
    while(i < n) {
        statement-set-2;
        for(int j = 0; j < m; j++) {
            statement-set-3;
        }
        statement-set-4;
        i++;
    }
} // scope for i
statement-set-5;
```

(b) [3.5 points] Replace both `for` loops with `while` loops

```
int m, n; cin >> m >> n;
statement-set-1;
{
    int i = 0;
    while (i < n) {
        statement-set-2;
        {
            int j = 0;
            while (j < m) {
                statement-set-3;
                j++;
            }
        } // scope for j
        statement-set-4;
        i++;
    }
}  // scope for i
statement-set-5;
```

(c) [3.5 points] Replace both `for` loops with `do-while` loops

```
int m, n; cin >> m >> n;
statement-set-1;
{
    int i = 0;
    do {
        statement-set-2;
        {
            int j = 0;
            do {
                statement-set-3;
                j++;
            } while (j < m);
        } // scope for j
        statement-set-4;
        i++;
    } while (i < n);
} // scope for i
statement-set-5;
```

(d) [3.5 points] Replace the outer `for` loop with a `do-while` loop, and the inner `for` loop with a `while` loop.

```
int m, n; cin >> m >> n;
statement-set-1;
{
    int i = 0;
    do {
        statement-set-2;
        {
            int j = 0;
            while (j < m) {
                statement-set-3;
                j++;
            }
        } // scope for j
        statement-set-4;
        i++;
    } while (i < n);
} // scope for i
statement-set-5;
```

(e) [6 points] Use a single `for` loop

```
int m, n; cin >> m >> n;
statement-set-1;
for (int i = 0; i < m * n; i++) {
    if (i % m == 0)  {
        statement-set-2;
    }
    statement-set-3;
    if ((i + 1) % m == 0)  {
        statement-set-4;
    }
}
statement-set-5;
```

8. [3 points] Choose ALL the correct option/options to fill in the blanks given in the following code snippet to swap the values of two variables `x` and `y`

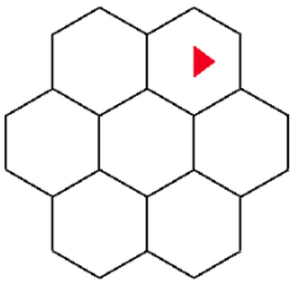
```
void swap(______(blank A)______) {
    ______(blank B)______;
    temp = a;
    a = b;
    b = temp;
}
main_program {
    int x = 0, y = 1;
    swap(______(blank C)______);
    cout << x << y;
}
```

Answer: b

```
(a) blank A: int a, int b
    blank B: int temp
    blank C: x, y

(b) blank A: int &a, int &b
    blank B: int temp
    blank C: x, y

(c) blank A: int *a, int *b
    blank B: int *temp
    blank C: &x, &y

(d) blank A: int *a, int *b
    blank B: int temp
    blank C: &x, &y
```

9. [10 points]

Complete the following code to draw the honeycomb consisting of a set of hexagons, as shown in the figure to the right. The code given in the box intends to draw a single hexagon of side length 50. The initial and the final position of the turtle will be at the center of the hexagon facing right. The start position of the turtle is given in the figure, and you are expected to draw the hexagons in the clockwise direction.



2 points for each correct answer

```
main_program {
    turtleSim();
    penUp();
    int i = 1;
    repeat(6) {
        left(90);
        forward(50);
        penDown();
        right(120);
        repeat(6) {
                forward(50);

                right(_____ 60 _____);
        }
        right(60);
        penUp();
        forward(50);

        left(_____ 90 _____);

        right(_____ 60*i _____);

        forward(50*sqrt(3));

        left(_____ 60*i _____);

        i = _____ i+1 _____;
    }
}
```

10. [12 points] Go through the program given below and answer the questions.

```
bool compare1 (int a, int *x) {
    cout << &a << " " << x << endl;
    return &a == x;
}
bool compare2 (int &a, int *x) {
    cout << &a << " " << x << endl;
    return &a == x;
}
bool compare3 (int &a, int *x) {
    cout << a << " " << *x << endl;
    return a == *x;
}
```

```
main_program {
    int a = 5;
    bool m = compare1(a, &a);
    bool n = compare2(a, &a);
    bool p = compare3(a, &a);
    bool q = m || n || p;
    bool r = m && n && p;
    cout << m << " " << n << " "
         << p << " " << q << " "
         << r << endl;
    return 0;
}
```

Assume that the address of variable `a` is `0x3fffd0`. You can assume other addresses if required in similar lines i.e. starting with `0x`. What is the output printed in: 3 points for each correct answer. The first address in (a) compare 1, can be anything, apart from 0x3fffd0, but should begin with 0x

(a) `compare1` 0x3fffd4 0x3fffd0

(b) `compare2` 0x3fffd0 0x3fffd0

(c) `compare3` 5 5

(d) `main_program` 0 1 1 1 0