A	# CS101 Spring 2020 Mid-semester Exam 11 questions, 12 pages, 50 marks 26th February 2020, 8:30 am to 10:30 am

Name: ____________________________ Roll number: __________________________

Lab No (L1/L2/L3): ___________ Group No.: _________ Department: _____________

You MUST write your final answer on this question paper itself. You can use spare pages on the question paper for rough work. Answers must be written in ***pen*** *(NOT pencil). Write your roll number on all pages. Assume all codes have the "simplecpp" header file included.*

(FOR GRADING PURPOSES ONLY)

Q #	Marks	Grading TA	Verifying TA	TA/Student Remarks
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				

Total: ____________________

Roll number: ___________________________

For Rough work

Roll number: ____________________________

Q1 (5 marks). The 52 cards in the playing deck are numbered from 1 to 52 as follows. Group the cards by their suits. The order of suits is Spades, Clubs, Hearts and Diamonds. Within a suit, order the cards in the increasing order of their value, Ace being counted as 1. Then number the cards as per this ordering.

For example: 3 of Spades gets the value 3, while 7 of Hearts gets the value 33.

The following code takes as input an integer value representing a card in the given fashion. Complete the code to print the card based on its value. Print the name for the face cards, which are Jack, Queen, and King. For others, print their numerical value.

```
#include <simplecpp>

main_program {
    int n;
    cin >> n;

    switch (___(i)___) {
        case ___(ii)___:
            cout << "King of "; break;
        case ___(iii)___:
            cout << "Queen of "; break;
        case ___(iv)___:
            cout << "Jack of "; break;
        default:
            cout << ___(v)___ << " of ";
    }

    switch (___(vi)___) {
        case ___(vii)___:
            cout << "Spades"; break;
        case ___(viii)___:
            cout << "Clubs"; break;
        case ___(ix)___:
            cout << "Hearts"; break;
        case ___(x)___:
            cout << "Diamonds";
    }
}
```

Answers:

(i) ______________

(ii) ______________

(iii) ______________

(iv) ______________

(v) ______________

(vi) ______________

(vii) ______________

(viii) ______________

(ix) ______________

(x) ______________

Ans: (i) n%13 (ii) 0 (iii) 12 (iv) 11 (v) n%13 (vi) (n-1)/13 (vii) 0 (viii) 1 (ix) 2 (x) 3

Grading: 0.5 marks per correct blank

Roll number: ____________________________

Explanation for (vi): As the numbering starts from 1, the suit is decided by (n-1)/13, not n/13.

Q2 (4 marks). The following program accepts 5 numbers and their respective 5 choices (`‘s’` or `‘f’`). If the choice is `‘s’`, then the program should print the sum of the digits of that number; whereas if the choice is `‘f’`, then the program should calculate the factorial of the number and print the sum of the digits of the factorial obtained. The function **‘fact’** calculates the factorial while the function **‘sum’** calculates the sum of the digits. Fill in the blanks to complete the program.

```
int sum(int n){
    int foo = 0;
    while(n!=0) {
       foo += _________(i)__________;
       n = n / 10;
    }
    return foo;
}
int fact(int n) {
    int bar = 1;
    while(n >= 1) {
        bar = bar * n;
        n--;
    }
    return _______(ii)________;
}
main_program {
    int n;    char choice;
    for (int i=1; i<=5; i++) {
        cin >> n;      cin >> choice;
        switch(choice) {
            case 'f':
                n = _____(iii)_____;
            case 's':
                n = _____(iv)_____;
        }
        cout << n << “\n”;
    }
```

Answers

(i) ____________________

(ii) ____________________

(iii) ____________________

(iv) ____________________

Roll number: ___________________________

}	

Ans: (i) n%10 (ii) bar (iii) fact(n) (iv) sum(n)

Grading: 1 mark per blank

Explanation: In a switch case, if break is not used, the code falls through the next cases till it encounters a break. Hence, if choice==’f’, two instructions will be executed. n = fact(n) and then n = sum(n). Hence, it effectively computes n = sum(fact(n)).

Q3 (3 marks). The following code reads 10 characters, one by one, and converts all lowercase alphabet to uppercase. It doesn’t change other characters. Fill in the blanks to complete the code.

Note: In the ASCII mapping, lowercase characters are mapped sequentially i.e. one after the another. Similarly, for uppercase characters. Note that the difference between ASCII of lowercase ‘a’ and uppercase ‘A’ is not 26.

```
main_program {
    char c;
    repeat(10) {
        cin >> c;
        if (_____(i)_____ <= c &&
                c <= _____(ii)_____) {
            c += _____(iii)_____;
        }
        cout << c;
    }
}
```

Answers.

(i) ____________________

(ii) ____________________

(iii) ____________________

Ans: (i) ‘a’ (ii) ‘z’ (iii) ‘A’ - ‘a’
Alternate solution: (i) 97 (ii) 122 (iii) -32.
Similarly, some characters may be replaced by their ASCII codes, hence there are 8 possible combinations.

Grading: 1 mark per blank

Q4 (4 marks). C++ has bitwise operations, which act on the binary representations of their operands bit by bit. “&, |, ~” are bitwise AND, OR and NOT respectively. “`a >> b`” shifts the bits of “`a`” to the right by “`b`” places.

Predict the output of the following instructions. The integer variable “`a`” is initialized to `90`.

Roll number: ___________________________

Code	Output
`cout << (a >> 2);`	
`cout << (a & 34);`	
`cout << (a | 45);`	
`cout << (a & (~42));`	

Ans: 22, 2, 127, 80

Grading: 1 mark per correct value

Explanation: C++ outputs the decimal representation of the number obtained after applying the bitwise operation.

Q5. (4 Marks) Given below are four programs. In each case predict the output.

<table>
<tr>
<td>

Program - i

```
void incr(int &a) {
        a++;
}

main_program {
        int a = 10;
        incr(a);
        cout << a;
}
```

</td>
<td>

Program - ii

```
void incr(int b) {
        int a = b;
        a++;
}

main_program {
        int a = 10;
        incr(a);
        cout << a;
}
```

</td>
</tr>
</table>

Roll number: ___________________________

Program - iii	**Program - iv**

`int incr(int a) {`
`        int b = a;`
`        b++;`
`        return a;`
`}`
 	
`main_program {`
`       int a = 10;`
`       a = incr(a);`
`       cout << a;`
`}`
	`int  incr(int b) {`
	`        int a = b;`
	`        a ++;`
	`        return a;`
	`}`
	
	`main_program {`
	`        int a = 10;`
	`        a = incr(a);`
	`        cout << a;`
	`}`

(i) ________________ (ii) ________________

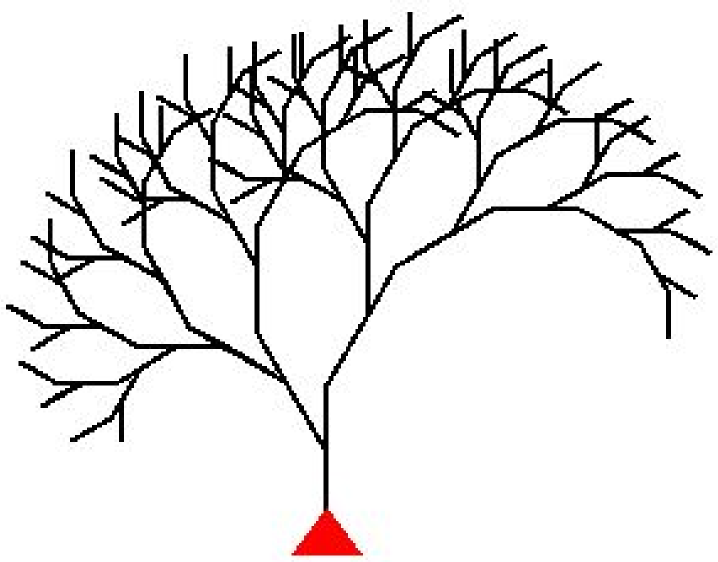
(iii) ________________ (iv) __________________

Ans: (i) 11 (ii) 10 (iii) 10 (iv) 11

Grading: 1 mark per blank

Q6 (4 marks). The program accompanying the following diagram for drawing a tree is given below. It uses a recursive approach for drawing such a tree. Fill in the blank spaces so that the program when executed draws the following diagram.

Roll number: ___________________________



```
#include <simplecpp>

void tree(int L, double trunkLength) {
      if (L > 0) {
            forward(____(i)____);
            left(30);
            tree(L-1, 0.8*trunkLength);
            right(____(ii)____);

            forward(0.4*trunkLength);
            right(____(iii)_____);
            tree(L-1, 0.8*trunkLength);
            left(30);

            forward(-trunkLength);
      }
}

main_program {
      turtleSim();
      left(____(iv)____);
      tree(7, 70);
}
```

Answers.

(i) __________________

(ii) __________________

(iii) __________________

(iv) __________________

Ans: (i)0.6*trunkLength (ii) 30 (iii) 30 (iv) 90

Alternate solutions for (ii), (iii), (iv). These values are angles so any angle which translates to the same value will also work. e.g. 30 is the same as -270 or 390.

Roll number: ___________________________

Explanation for (i): As the code moves forward by 0.4*trunkLength in line 10, and then backwards by trunkLength, in line 15, the value in (i) must be so that the turtle gets back to the original position. Hence the answer is 0.6*trunkLength. Similar explanations apply to (ii) and (iii).

Grading: 1 mark per blank

Q7. (6 marks) Consider the functions f and g. What will be the value of “a”, “b” and “c” after the two main_program segments (Program 1 and Program 2) are executed:

```
bool f(int* x, int* y){
    *x -= 8;
    int temp;
    temp = *x;
    *x = *y;
    *y = temp;
    return true;
}

bool g(int *y){
    *y += 20;
    if(*y > 100){
          return false;
    }
    else return true;
}
```

Program Segment 1	Program Segment 2

`main_program{`
`   int a = 5, b = 89;`
`   bool c = g(&b) || f(&a, &b);`
`}`
	`main_program{`
	`   int a = 5, b = 89;`
	`   bool c = g(&b) && f(&a, &b);`
	`}`

What would be the value of a, b and c? **a** ______________________ **b** ______________________ **c** ______________________	What would be the value of a, b and c? **a** ______________________ **b** ______________________ **c** ______________________

Ans: a 109, b -3, c true, a 5, b 109, c false

Grading: 1 mark per blank

Roll number: ___________________________

Explanation: For OR (i.e. ||), if the first expression evaluates to TRUE, then the subsequent expressions are not evaluated, as the answer is known now. Similarly for AND (i.e. &&), if the first expression is FALSE, then the subsequent expression is not evaluated.

Q8 (6 marks). What is the output printed by the following program?

```
main_program{
      int a=5, b=8, c=13;
      int *p = &a, *q = &b, *r=&c;
      int count = 2;
      while(count >0){
            p = q; q = r; r = p;
            count--;
      }
      cout << *p << " "<< *q << " " << *r << endl;
      cout << a << " " << b << " " << c << endl;

      count = 2;
      while(count > 0){
            *p = *q; *q = *r; *r = *p;
            count --;
      }
      cout << *p << " "<< *q << " " << *r << endl;
      cout << a << " " << b << " " << c << endl;
}
```

Please provide your output here:

(i) First output line: ______ ______ ______

(ii) Second output line: ______ ______ ______

(iii) Third output line: ______ ______ ______

(iv) Fourth output line: ______ ______ ______

Ans: 13 8 13

5 8 13

8 8 8

5 8 8

Grading: 0.5 mark per blank

Roll number: ____________________________

Q9 (6 marks). Linear diophantine equation for a tuple <`a, b`> is "`ax + by = 1`" where "`a`" and "`b`" are given positive integers, and the variables are "`x`" and "`y`". A solution to this equation is integer assignments to the variables, such that the equation is satisfied (note that variables can take negative integer values).

For example, "`7x + 5y = 1`" has as a solution "`x = -2, y = 3`". Note that some equations may have no solutions, such as "`12x + 16y = 1`".

A recursive way to find a solution, if one exists, is to do the following.

I. In the base case: If "`a`" is divisible by "`b`":
   A. If "`b`" is 1, then the solution is "`x = 0, y = 1`".
   B. Otherwise, there is no solution.
II. Otherwise, find the solutions "x_1, y_1" for the tuple <`b, r`>, (`r` is the remainder when "`a`" is divided by "`b`"), if one exists. Then do the following:
   A. $x = y_1$
   B. $y = x_1 - q * y_1$, where "`q`" is the quotient when "`a`" is divided by "`b`".

Complete the following code which implements this recursive algorithm.

```
#include <simplecpp>

bool diophantine(int a, int b, int &x, int &y){
    if (_____(i)_____) {
        if (_____(ii)_____) {
            x = 0;
            y = 1;
            return true;
        } else {
            return false;
        }
    }

    if (diophantine(_____(iii)_____)) {
        int u = x;
        int v = y;
        int q = _____(iv)_____;
        x = _____(v)_____;
        y = _____(vi)_____;
        return true;
    } else {
        return false;
    }
}
```

Answers.

(i) ______________

(ii) ______________

(iii) ______________

(iv) ______________

(v) ______________

(vi) ______________

Roll number: ___________________________

Answers: (i) a % b == 0 (ii) b == 1 (iii) b, a % b, x, y (iv) a / b (v) v (vi) u - q*x

Grading: 1 mark per blank

Q10 (5 marks). The function `eulerTotient`, implemented below, calculates the Euler's Totient function for a positive integer "`n`", which it takes as its argument. The totient function is defined as

$$\phi(n) = n \prod_{p\,|\,n} (1 - \tfrac{1}{p})$$

where the product is over all primes that divide "`n`". But the code has errors in exactly 5 lines, some syntax errors, some logical. In the table, fill the line numbers which have errors, and provide correct code for these lines. A line may have multiple errors.

```
 1 void isPrime(int n) {
 2     int m = n-1;
 3     for (int i=2; i<=m; i++) {
 4         if (n % i == 2) return false;
 5     }
 6     return true;
 7 }
 8
 9 int eulerTotient(n) {
10     int phi = n;
11     for (int i=2; i<=n; i++) {
12         if (n%i == 0 && isPrime(i)) {
13             phi = (phi / (i+1)) * (i-1);
14         }
15     }
16     return;
17 }
```

Line number	Correct code

Answers

Line number	Correct code

Roll number: ___________________________

1	`bool isPrime(int n)`
4	`if (n % i == 0) return false;`
9	`int eulerTotient(int n) {`
13	`phi = (phi / i) * (i-1);`
16	`return phi;`

Alternate solution for Line 1: int isPrime(int n)
Alternate solution for Line 9: unsigned int eulerTotient(unsigned int n) (i.e. someone could use unsigned int for either the argument or the return value or both).

Grading: 1 mark if both the line number and the provided code is correct (no partial marking)

Q11. (3 marks) The following recursive program reverses a sequence of integers. It first takes the length of the sequence as input n. Then it takes n integer inputs. The output is the reverse sequence (with spaces between the integers).

Example:
Sample input:

```
4
45 78 99 23
```

Sample Output:

```
23 99 78 45
```

```
#include <simplecpp>

void reverse(int n)
{
    int x;
    cin >> x;
    if(n == 1){ // base case
        _____ (i)______;
    }
    else{ // recursive part
        ______(ii)_______;

        ______(iii)______;
    }
}

main_program{
     int n;
     cin >> n;
```
	Answers:
	
	(i) ______________________________
	
	(ii) ______________________________
	
	(iii) ______________________________

`    reverse(n);`
`}`	

Ans: (i) cout << x << “ “; (ii) reverse(n-1); (iii) cout << x << “ “;
Alternate solution: (i) cout << x; (iii) cout << x;
Alternate solution for (ii) reverse(--n) (Note: reverse(n--) won’t work correctly).

It’s okay if the provided solution doesn’t print spaces.

Grading: 1 mark per blank