CS101: Autumn 2017 Mid Semester Exam

15th September 2017, 3pm to 5pm

Name: _____**Umesh Bellur**_____

Roll number: _____ Hall/Column_____

Lab Group: (Eg: Wed-SL2-2)_____

Write all your answers ONLY in the space provided. Do not write irrelevant answers. Answers must be written in pen (not pencil) ONLY. Write your roll number on all pages including any extra sheet you may use. Any minor syntax errors in code snippets are typos - ignore them and assume the right syntax.

DO NOT WRITE IN THE TABLE BELOW

Q #	Marks	Grading TA	Remarks
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			

CS101: Autumn 2017 Mid Semester Exam

15th September 2017, 3pm to 5pm

Name: _____ **Roll number:** _____

1. The following program is meant to print **5 ! = 120** but has a mistake. Write the correct version of the (only) ONE line of code that needs to be fixed in the box on the right. **[2 points]**

```
int main() {
  int N, total;
  total = 1;
  cin >> N;
  while (N > 1) {
    total *= --N;
  }
  cout << total;
  return 0;
}
```

Ans:

```
total *= N--;

OR

total *= N; N = N-1;
```

No partial points.

2. Convert the **while** loop in the previous problem to the following **for** loop by filling in the blanks. You may ***not*** declare any other variables or change any of the given code. **[3 points]**

```
int main(){
  int N, total;
  total = 1;                                  // 1 point each. Note the
  for (cin >> N; N > 1;          ) {          // last blank MUST be empty.
    total *= theNum;
     N = N - 1;
  }
  cout << total;
  return 0;
```

3. Write down the values for all variables appearing in each statement. **[5 points]**

```
int main(){
  int x, y, z;
  float a, b;
  x = 3;  y = 5.3;  z = 2;
  a = 2.5; b = 3.14;
/*1*/   x += y + a * z-b;     x  = 9

/*2*/   a = b / (x%y + z);    a _= 0.523_ OR 0.52_ or 3.14/6

        z += (x == y);        z = 2__

/*3*/   x == (a = b);         x  = 9_
      return 0;
}
```

Rewrite the statement labeled /*1*/ with the correct braces so that X is assigned the value 14.

CS101: Autumn 2017 Mid Semester Exam

15th September 2017, 3pm to 5pm

Name: _____ **Roll number:** _____

_____x += (y+a)*z-b;_ 1 point for each blank. _____

4. Predict the output of the following program assuming ascii characters are in sequential order. **[3 points]**

```
int main() {
  char c;
  for (c = 'a'; c < 'g'; ++c) {
    switch (c){
      case 'a':  c += 2;
      case 'c':  c += 1;
      case 'g':  ++c;
          cout << c-- << endl;
      default:  ++c;
    }
    cout << c << " ";
  }
  return 0;
}
```

ANSWER: _____ e e g 1 point per letter_____

5. You have an integer variable x that holds a four-digit positive integer. Write a **single assignment expression** to store the number formed by middle two digits in integer variable y. **[3 points]**

ANSWER: **y = x / 10 % 100; -OR- y = x % 1000 / 10; OR**
y = ((x%1000)-(x%10))/10 NO PARTIAL POINTS

6. What value will the fragment of code shown below print?____**13 - no partial points)** **[2 points]**

```
int *iptr;
int i, arr[2][2] = {10, 11, 12, 13};
iptr = *arr ;
cout << *(iptr+3);
```

7. Examine the code below. In one short sentence, what (not how) does the following function do? **[2 points]**

```
int mystery(int x, int y, int z) {
      int temp;
      temp = x;
      if(y > temp) { temp = y; }
```

ANSWER: **It returns the maximum of x, y and z. No partial points for any other explanation.**

CS101: Autumn 2017 Mid Semester Exam

15th September 2017, 3pm to 5pm

Name: _____ **Roll number:** _____

`if(z > temp) { temp = z; }`
`return temp;`
`}`	

8. Write down the DECLARATION only of a *function* called **RectangleProperties** that computes and returns the area and perimeter of the rectangle based on the height and width of the rectangle (which are double precision floating point numbers). Use of Structs is NOT permitted here. **[2 points]**

```
void RectangleProperties(double height, double width,
                         double& area, double& perim);
                    OR
double RectangleProperties(double height, double width,
                         double& area); // or double& perim
```

0.5 point for return type, 0.5 point for the height and width parameters and 0.5 point each for the reference parameters. Its NOT FLOAT, its double.

9. Assume **A** is an array of **N (>2)** integers. Write the expression for "find the sum of the first and last entries and assign it to the third element" using array indexing notation. **[2 points]**

ANSWER: `A[2] = A[0] + A[N-1];` ***RHS=1 pt and LHS = 1pt.***

10. State the output of the following code fragment:___cat_ **[2 points]** **no partial points.**

```
char *arr[] = { "ant", "bat", "cat", "dog", "egg", "fly" };
char** ptr = arr;
char *ptr1 = (ptr += sizeof(int))[-2];
cout << ptr1;
```

11. Here is another fragment of code. Circle your choice as the output of the fragment in the box on the right. **[1 point]**

`int x = 5;`
`const int* ptr = &x;`
`*ptr = 10;`
`cout << x ;`
	1. **Compilation error**
	2. 10
	3. 5
	4. Garbage value

CS101: Autumn 2017 Mid Semester Exam

15th September 2017, 3pm to 5pm

Name: _____ **Roll number:** _____

12. Here is a function that concatenates the src string ***to*** the dest string. Fill in the blanks with one line of code each. The function is to be written using ONLY pointer arithmetic and you are NOT allowed to use the array indexing operator. **[4 points]** **1 point for each blank.**

```
void strcat(char* dest, char* src) {
 char *p,*q;
 // First, make p point to end of dest
 p=dest;
 while(*p) { __p++_____; }
 q=src;
 //copy from *q to *p

 while(___*q_____){

  ___*p = *q___;
  p++;
  q++;
 }
  ___*p = '\0'__; // terminate the result.;
}
```

CS101: Autumn 2017 Mid Semester Exam

15th September 2017, 3pm to 5pm

Name: _____ **Roll number:** _____

13. RECURSION - Fill in the blanks in the shown ***RECURSIVE*** program that prints all binary strings of a given size n.
For example for n=3 it will print **000 100 010 110 001 101 011 111** *in order.*
For n=4 it will print **0000 1000 0100 1100 0010 1010 0110 1110 0001 1001 0101 1101 0011 1011 0111 1111** *in order. And so on.* **[6 points]**

```
void generateBinary(int length, char* string){
        if(length > 0){

                string[_length-1___] = '0';  // 0.5 + 0.5

                generateBinary(length-1, string);      // 1

                string[length-1] = '1';                // 1.5

                generateBinary(length-1, string); // 1.5

        } else
                cout << string << "  ";
}

int main(){
        int n;
        cin >> n;

        char s[n+1];    // 0.5 point

        s[n] = '\0';    // 0.5 point

        generateBinary(n,s);
        return 0;
}
```

Name: _____ **Roll number:** _____

14. Rotating a matrix. One by one rotate all rings of elements, starting from the outermost. To rotate a ring, we need to do following.

1) Move elements of top row one column right. The last element comes down one row.
2) Move elements of last column one row down. The last element moves left on the last row.
3) Move elements of bottom row one column left
4) Move elements of first column one row up.

Repeat above steps for inner ring while there is an inner ring. Examples of 3x3 and 4x4 matrices are shown below.

For a 3x3 Matrix **For a 4x4 matrix**

Input	**Output:**	**Input:**	**Output:**
1 2 3 4 5 6 7 8 9	4 1 2 7 5 3 8 9 6	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	5 1 2 3 9 10 6 4 13 11 7 8 14 15 16 12

Fill in the blanks for the given program to rotate a `mxn` matrix. **[8 points]**

CS101: Autumn 2017 Mid Semester Exam

15th September 2017, 3pm to 5pm

Name: _____ **Roll number:** _____

```
const int R = 5;     const int C = 5;

// A function to rotate a matrix mat[][] of
//  size R x C.
// Initially, m = R and n = C

void RotateMatrix(int m, int n,
                  int mat[R][C]) {
    int row = 0, col = 0;
    int prev, curr;
  /*  row - Starting row index
     m - ending row index
     col - starting column index
     n - ending column index
     i - iterator
  */                          // 3 points
  while (row < m && col < n)  {
     // Store the first element of next row, this
     // element will replace first element of
     // current row

         prev =mat[row+1][col]; //0.5

     /* Move elements of first row from the
        remaining rows */
    for (int i = col; i < n; i++) {
            curr = mat[row][i];

           mat[row][i] = prev; //0.75
            prev = curr;
        }
        row++;

     /* Move elements of last column from the
     remaining columns */
      for (int i = row; i < m; i++)  {

            curr = mat[i][n-1]; // 0.75
            mat[i][n-1] = prev;
            prev = curr;
        }
        n--;

    /* Move elements of last row from the
      remaining rows */
      if (row < m) {
          for (int i = n-1; i >= col; i--)
          {
              curr = mat[m-1][i];

              mat[m-1][i]= prev; // 0.5x2

              prev = curr;
           }
       }
       m--;

     /* Move elements of first column from the
        remaining rows */
     if (col < n) {
         for (int i = m-1; i >= row; i--)
         {
             curr = mat[i][col];
              mat[i][col] = prev; // 2
             prev = curr;
          }
      }
      col++;
   } // End of While loop
} // End of Function
```

CS101: Autumn 2017 Mid Semester Exam

15th September 2017, 3pm to 5pm

Name: _____ **Roll number:** _____

15. **STRUCTURES** - A stack is a data structure that obeys Last In First Out property. What this means is that the last element we ***pushed*** on to the stack is at the ***top*** of the stack and is the element that is removed when we ***pop*** the stack. Given that we want to create a stack of **int,** here is the definition of a struct in C++ to do this:

```
const int size = 20;
struct Stack {
      int data[SIZE];
      int top;  // maintains the index in the array that represents the top of the stack.
                // The top meaning the last element pushed on the stack.
               // Assume that top is initialized to 0 when a stack instance is declared.
 };
```

Augment this with a **member function - *push*** and a normal (non-member) function ***pop*** with the following specifications:

[3 points] *push* - takes an integer to be pushed and puts in on top of the stack if the stack is not full. If the stack is full it does nothing. Remember that a stack may contain some number of elements already**.** The new integer being pushed must not overwrite any of the existing elements.

[3 points] *pop* - removes the element at the top of the stack and resets top to the element just above it. If the stack is empty, pop does nothing.

Each member function may contain ONE if statement with the corresponding else if needed but can contain only ONE code statement as the action of the IF or ELSE branch. No other code.

```
struct Stack{
      int data[SIZE];
      int top;
      void push(int x){                    // 1.5 points for the sig
           if(top < (SIZE-1))              // 1.5 points for the code
                data[++top] = x;
      }
};

void pop(Stack& s){                  // 1.5 points for the sig
     if(s.top > 0)                   // 1.5 points for the code
          s.top--;
     }
}
```

CS101: Autumn 2017 Mid Semester Exam

15th September 2017, 3pm to 5pm

Name: _____ **Roll number:** _____

16. In the context of a 2D point structure with x and y coordinates being double precision floating point values, the function CenterOfGravity finds the center of gravity of an array of n Points. Fill up the missing code Below. You MUST do pointer based access to the elements of the struct array - you may NOT use array indexing to access elements of pts in the function. **[6 points]**:

// Declare the structure here to represent a 2D point based on the code below.

```
struct Point { double x, y; };    // 2 points
```

// Function to compute CG Point

```
Point CenterOfGravity( Point *pts, unsigned int n) {
      unsigned int i;

      Point cg = {0,0}; // Intialize cg    // 1 point

      // Compute the sum of x and y coordinates respectively

      for(i = 0; i < n; ++i, ++pts) {     // 0.5 points
            cg.x += pts->x;               // 0.5 points
            cg.y += pts->y;               // 0.5 points
      }   // OR leave the first one empty and do pts++->x and pts++->y
      // Normalize the sum by the number of points to find CG

      cg.x  /= n;  // 0.5 for both put together

      cg.y  /= n;

      return cg;
}

int main() {
      Point points[] = {2, 5, 7, 5, 3, 6, 4, 4}; // works fine, dont worry.
      // Compute CG for points[]
      Point cg = CenterOfGravity(points, 4);  // 1 point
      cout << cg.x << "\t"  << cg.y << endl;
}
```