Indian Institute of Technology, Delhi

FALL, 2015

COL 100: INTRODUCTION TO PROGRAMMING

Minor 2

One Hour

NOTE: Total Marks: 40 **Total Number of Pages : 10**

Name: [redacted]	
Group No: 15	Entry No: 2015[redacted]

Marks:

1	2	3	4 H	5	6 SG	7 VM	8 S	Total
2	0	0 mk	2.5	$5\frac{1}{2}$ SS	4	3	4	21

1. Read the following instructions and indicate which one you like best. (2 marks)

(a) All answers need to be brief and to the point.
(b) Please make any assumptions that you deem to be reasonable.
(c) Follow the *spirit of the question*. Do not immerse yourself in irrelevant details.
(d) Every answer needs to be written neatly and cleanly in the space provided for it.
(e) Do your rough work seperately and write only your final code in the question.
(f) Use proper handwriting, and do not write anything on the margins.
(g) Calculators and mobile phones are **not permitted**.
(h) The closer your answer is to the model answer in terms of the lines of code, the more marks you get.
(i) The final answer needs to be written with a pen.
(j) There are two additional pages at the end for rough work.
(k) This question paper **NEEDS TO BE SUBMITTED**. Do not take it with you.

2

The best instruction is: g

2. Consider the array: [79, 43, 31, 96, 64, 30, 80, 26, 1]. Sort this array using the merge sort algorithm. Show the contents of the subarrays after each call to the *merge* routine.

(7 marks)

3. We want to write a function that returns the 19th bit in an integer. The least significant bit is the first bit (rightmost bit), and the most significant bit is the 32nd bit. Complete the function listed below. Use the variables, i and j, if required. The model answer contains 4 C statements.

(4 marks)

```
int return19thbit (int x){
        int i=0,j=1;

    ~~for (      i<  x/%/j/;~~

      for (i=9 ; i<=18 ; i++)  {
                x = x/j ;
                    }

        x = x
}
```

0

4. Complete the following **recursive function** to reverse an array. Example: array is {1,2,3,4,5}. The reversed array is: {5,4,3,2,1}. You can assume a function, swap(int *a, int *b), that swaps two integers. The model answer contains 3 C statements.

(6 marks)

2½

```
void array_reverse(int values[], int startIdx, int endIdx) {

    startIdx =0 ;   ~~endIdx=0;~~

for ( startIdx =0 ; startIdx <= endidx ,  startidx ++) {

            endidx = 4-startidx;

      swap ( &a values [startidx ) , & values [endidx) ) ;
```

5. Given the *current time* and *time* required for a job, it is desired to find out the *time* when the job gets completed. Write a structure to represent time in hours, minutes and seconds, and write a function which gets the *current time*, and *job time*, and returns the completion *time*.

(6 marks)

```
struct time {

    int hour ;
    int min ;
    int sec ; } ;   current , job ; complete ;

struct time function ( struct time current , struct time job ) { ,
                                           struct time complete)

        int   seconds , minutes, hours ;
        int   temp ; temp2 ;temp 3 ;

  seconds  =  (current) . sec + (job) . sec ;
  seconds  =  seconds % 60 ;      temp1= seconds /60 ;

  minutes  =  current . min + job. min + temp1 ;
  minutes  =  minutes % 60 ;      temp 2 = minutes/60 ;

  hours  =  current . hour + job. hour + temp 2 ;
  temp 3 =  hours /24 ; .
  hours  =  hours - 24 * temp 3 ;

  complete . sec = seconds ;
  complete min = minutes ;
  complete . hour = hours ;

    return complete ;
```

1

2

1 1/2

1

5½

6. The following functions are supposed to raise a number m to power n. Correct the functions if needed. Discuss which of the two functions is more efficient. State your reasons.

(4 marks)

```
int raise (int m, int n)
{
  if ( n==1) return m;
  else   {
    if (n%2 == 0) return  raise( m, n/2)*raise(m, n/2);
    else return  m*raise( m, n/2)*raise(m, n/2);
  }
}

int raise2 (int m, int n)
{
  int b;
  if ( n==1) return m;
  else {
    b = raise2(m, n/2);
    if (n &   1 == 0) return  b*b;
    else return  m* b*b;
  }
}
```

4

2nd is more efficient. Because there are more no. of calls in 'raise' ~~gri~~ resulting in greater amounts of computation.

7. What is the output of following program. (5 marks) 6

```
#include <stdio.h>
struct complex {
  int real;
  int imag;
};

void increase(struct complex *t, struct complex *s) {
  struct complex r = *t;
  *t = r;
  *t = *s;
  *s = r;  }

int main() {
  int i; struct complex A[5], *p;
  A[0].real = 8; A[0].imag = 6;
  A[1].real = 3; A[1].imag = 5;
  A[2].real = 4; A[2].imag = 2;
  A[3].real = 1; A[3].imag = 1;
  A[4].real = 1; A[4].imag = -1;

  p = A;
  printf("%d  %d  %d\n", *(p++).real,  ++p->imag, *(p+1).imag);
  increase(A, A+1);
  printf("%d  %d  ", A[0].real,  A[0].imag);
}
```

3

8 6 3 5 2
----

3 5
----

8. Complete the following program to print 4 different components of the string, *ipaddr*. The final output should be: 199 201 26 324. You **cannot** add any more loops or function calls (other than *atoi*). You must generate *components[...]* so that this can be used in the *printf* statement given below. Note that a closing '}' does not count as a C statement. You can use function *atoi()* to convert string to integer. The model answer contains 5 additional C statements.

(6 marks)

```
int main(){
        char ipaddr[] = "199.201.26.324";
        int length = strlen(ipaddr);
        int idx;
        int components[4];
        char *ptr = & ipaddr[0];
        int tempIdx = 0;

        for (idx = 0; idx < length; idx++){

              int temp=0;
              if ( ipaddr[idx) == '.') {
                    temp ++;
                    ~~atoi~~ ipaddr[idx] = \0;
                    component[temp-1] = atoi ( ipaddr); }

        }

        printf ("%d %d %d %d \n", components[0], components[1],
                              components[2], components[3]);
}
```

4