INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

Stamp / Signature of the Invigilator

EXAMINATION (End Semester)											**SEMESTER (Spring)**		
Roll Number										**Section**		**Name**	END-SEM SOLUTION (Spring 2018)
Subject Number	C	S	1	0	0	0	1	**Subject Name**				*Programming and Data Structures*	
Department / Center of the Student												**Additional sheets**	

Important Instructions and Guidelines for Students

1. You must occupy your seat as per the Examination Schedule/Sitting Plan.
2. Do not keep mobile phones or any similar electronic gadgets with you even in the switched off mode.
3. Loose papers, class notes, books or any such materials must not be in your possession, even if they are irrelevant to the subject you are taking examination.
4. Data book, codes, graph papers, relevant standard tables/charts or any other materials are allowed only when instructed by the paper-setter.
5. Use of instrument box, pencil box and non-programmable calculator is allowed during the examination. However, exchange of these items or any other papers (including question papers) is not permitted.
6. Write on both sides of the answer script and do not tear off any page. **Use last page(s) of the answer script for rough work.** Report to the invigilator if the answer script has torn or distorted page(s).
7. It is your responsibility to ensure that you have signed the Attendance Sheet. Keep your Admit Card/Identity Card on the desk for checking by the invigilator.
8. You may leave the examination hall for washroom or for drinking water for a very short period. Record your absence from the Examination Hall in the register provided. Smoking and the consumption of any kind of beverages are strictly prohibited inside the Examination Hall.
9. Do not leave the Examination Hall without submitting your answer script to the invigilator. **In any case, you are not allowed to take away the answer script with you.** After the completion of the examination, do not leave the seat until the invigilators collect all the answer scripts.
10. During the examination, either inside or outside the Examination Hall, gathering information from any kind of sources or exchanging information with others or any such attempt will be treated as **'unfair means'**. Do not adopt unfair means and do not indulge in unseemly behavior.

Violation of any of the above instructions may lead to severe punishment.

Signature of the Student

To be filled in by the examiner

Question Number	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Total**
Marks Obtained											

Marks obtained (in words)	**Signature of the Examiner**	**Signature of the Scrutineer**

Instructions related to this questions paper:

1. This is a QUESTION-CUM-ANSWER script.
2. This question paper has TEN questions.
3. Answer all questions.
4. All answers should be written in the blank spaces provided immediately after the question / filling in the blanks as specified. Q1, Q2 and Q9a are of multiple-choice type. Only one answer is correct, and the correct choice should be circled.
5. No clarifications! Make appropriate (but not unjustifiable) assumptions wherever (you feel) necessary.
6. Extra space is provided for rough work in the question paper itself. **No extra pages should be attached with this question-cum-answer script.**

1. Assume that on a certain machine an int variable is of size 32 bits (or 4 bytes), char variable is of size 8 bits (or 1 byte), and each memory address is of size 32 bits. Assume further that the sizeof() function call returns the size of its operand in bytes. Answer the following questions. **Marks:** $1+2=3$

   (a) Consider the following structure:

```
struct myStruct {
  char name[20];
  int account_number;
  struct myStruct *next;
};
```

   What does sizeof(struct myStruct) return on this machine?

A. 26	B. 28	C. 48	D. 46

   (b) What will be the output of the C program?

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int *p;
    p = (int *)malloc(20);
    printf("%d\n", sizeof(p));
    free(p);
    return 0;
}
```

A. 40	B. 20	C. 4	D. 80

2. Answer the following questions. **Marks:** $3 \times 2 = 6$

   (a) Consider the following sequence of push and pop operations on an initially empty stack S.

```
S = push(S,1);
S = pop(S);
S = push(S,2);
S = push(S,3);
S = pop(S);
S = push(S,4);
S = pop(S);
S = pop(S);
```

   Which of the following is the correct order in which elements are popped?

A. 1 2 4 3	B. 1 3 2 4	C. 1 2 3 4	D. 1 3 4 2

   (b) What is the output of the following program? **Marks:** 2

```
#include<stdio.h>
#include<stdlib.h>

int main()
{
    int *ptr;
    *ptr = 10;
    *ptr = 20;
    *ptr = 30;
```

```
    printf("%d\n",*ptr);
    return 0;
}
```

Actually this will result in an error as memory for the pointer has not been alloted.

A. 10	B. 20	C. 30	D. None of the above

(c) Under what condition will this program print the string "Hello"?

```
#include<stdio.h>
#include<stdlib.h>

int main()
{
    int *ptr;
    ptr = (int *)malloc(sizeof(int)*10);
    if (ptr == NULL)
        printf("Hello\n");
    return 0;
}
```

A. if the memory could not be allocated to the pointer "ptr"
B. if the memory has been allocated to the pointer "ptr" successfully
C. it will never print
D. none of the above

3. Fill in the blanks to complete a C program that creates a singly linked list by repeatedly calling the function push(). It then counts the number of nodes present in the singly linked list recursively using the function getCount(). Each blank has at most one statement.

```
// Recursive C program to find length or count of nodes in a linked list
#include<stdio.h>
#include<stdlib.h>

/* Link list node */
struct Node
{
    int data;
    struct Node* next;
};

/* Given a reference (pointer to pointer) to the head of a list and
   an int as parameters, the function pushes a new node on the front
   of the list. */
void push(struct Node** head_ref, int new_data)
{
    /* allocate node */
    struct Node* new_node =
            (__struct Node*__) malloc(__sizeof(struct Node)__);

    /* put in the data  */
    new_node->data  = new_data;

    /* link the old list off the new node */
    new_node->next = (*head_ref);
```

```
    /* move the head to point to the new node */
    _____(*head_ref) = new_node_____;

/* Counts the no. of occurences of a node
   (search_for) in a linked list (head)*/
int getCount(struct Node* head)
{
    // Base case
    if (_____head == NULL_____)
        return 0;

    // count is 1 + count of remaining list
    return 1 +_____getCount(head —> next)_____;
}

/* the main function*/
int main()
{
    /* Start with the empty list */
    struct Node* head = NULL;
    int num;
    char flag = 'Y';

    // Use push() repeatedly to construct the list
    while(____flag == 'Y' (OR) flag != 'N'____)
        {
        printf("\n Enter the next number to be pushed into the stack: ");
        scanf("%d", &num);
            push(_____&head_____, num);
        printf("\n Do you want to push more numbers into the stack?
        (Answer Y to continue and N to stop pushing into the stack)");
        scanf("%c", &flag);
        }

     /* printing the size of the list created */
    printf("\n Number of nodes in the linked list is %d",
    _____getCount(head)_____);
    return 0;
}
```

4. Answer the following questions. **Marks:** $3+4+2=9$

   (a) A quadratic algorithm with processing time $T(n) = cn^2$ spends 1 ms in processing 100 data items. Time spent for processing 5000 data items = __2500 ms (OR) 2.5 sec__

   (b) Consider the problem of exponentiation of integer x to the power of integer n (i.e., x^n). A straightforward way of doing this is to multiply x, n times. However a more efficient way to solve this problem would be to see that $x^n = x^{n/2} * x^{n/2}$. Assuming that $n = 2^k$, we can write a small recursive function to implement exponentiation.

```
int power(int x, int n){
    if (n==0) return 1;
    if(n==1) return x;
```

```
    if ((n % 2) == 0) return power(x*x, n/2);
}
```

Let the time required to execute this program be $T(n)$. Assume $T(0) = c_1$ and $T(1) = c_2$.

(i) The recursive expression is given by $T(n) =$ T(n/2) $+ c_3$

(ii) The exact solution to the above recursive expression is c2 + c3.log(n)

(c) $log(n!) = \Theta($ n $* log(n))$.

5. (a) What does the following function do on the elements of the array $arr[]$? **Marks:** 2

```
void whatdoIdo(int arr[], int size)
{
    int i=0;

    for(i=0; i < size; i++)
    {
        if( arr[i] % 2 == 0 )
            arr[i] = 0 ;
        else
            arr[i] = 1 ;
    }
}
```

The function whatdoIdo replaces even elements in the array with 0 and odd elements with 1.

(b) The following function computes the median of an array of floats $x[]$. Assume that all the entries in the array are distinct, and there is only a single digit after the decimal point for all the numbers. Fill the blanks.

Each blank can have only ONE statement.**Marks:** $0.5+0.5+0.5+0.5+0.5+0.5+1+1+1=6$

```
float median(int n, float x[]) {
    float temp;
    int i, j;
    for(i=0; i<n-1; i++) {
        for(j= i+1 ; j< n ; j++) {
            if( x[j] < x[i] ) {
                temp = x[i] ;
                x[i] = x[j] ;
                x[j] = temp ;
            }
        }
    }

    if( n%2 == 0 ) {
        return ( (x[n/2] + x[n/2 - 1]) / 2.0 ) ;
    } else {
        return x[n/2] ;
    }
}
```

(c) Now suppose you have two already sorted arrays ar1[] and ar2[] of EQUAL size n. The following function attempts to find the median of the elements of the two arrays combined together. For instance if ar1 = $\{1.0, 12.0, 15.0, 26.0, 28.0\}$ and ar2 = $\{2.0, 13.0, 17.0, 30.0, 45.0\}$, you have to find the median of the elements $\{1.0, 12.0, 15.0, 26.0, 28.0, 2.0, 13.0, 17.0, 30.0, 45.0\}$. Fill in the blanks. Each blank has only ONE statement.

Marks: $(0.5+1)+(0.5+1)+0.5+(0.5*3)+(0.5*3)+1.5=8$

```
/* This function returns median of ar1[] and ar2[].
           Assumptions in this function:
           Both ar1[] and ar2[] are sorted arrays
           Both have n elements */
float getMedian(float ar1[], float ar2[], int n)
{
    int i = 0;
    int j = 0;
    int count;
    float m1 = -1.0, m2 = -1.0; // contains medians from two arrays
    for (count = 0; count <= n; count++)
    {
        if (i == n)
        {
            m1 = _____m2_____;
            m2 = ____ar2[0]____;
            break;
        }
        else if (j == n)
        {
            m1 = _____m2_____;
            m2 = ____ar1[0]____;
            break;
        }
        if (ar1[i] __< ar2[j]__)
        {
            m1 = _____m2_____;
            m2 = ____ar1[i]____;
            i_____++_____;
        }
        else
        {
            m1 = _____m2_____;
            m2 = ____ar2[j]____;
            j_____++_____;
        }
    }
    return _____(m1 + m2)/2.0_____;
}
```

6. The following recursive function takes a string of given length as input and determines whether the string is a palindrome. It returns 0 if the string is not a palindrome and 1 if it is. Fill in the blanks. Each blank can have AT MOST one statement. **Marks:** $1+2+2=5$

```
int ispalindrome ( char A[], int n )
{
        if (__n <= 1__) return 1;/*base case*/
        if (A[0] != A[n-1]) return 0;
        return ispalindrome(__&A[1], n-2__);
}
```

7. Complete the following program, where the main function takes three strings A, B, C as input from the user and determines whether the string A contains the regular expression $B * C$, where $*$ stands for any substring. For instance, if A = "abcdefg", B = "bc" and C = "ef", the function determines if an occurrence of "bc" followed (not necessarily immediately) by an occurrence of "ef" can be detected in "abcdefg". In this case, the occurrence $B * C$ is detected at index position 1 in A, and the main function gives this output. Either of B or C can also be null. The main function makes use of another function *locateSubstr* that checks whether a string A contains another string B as a substring, and if so, returns the match index of B in A. Thus, when B and C are non-empty, the main function first finds if B is a substring of A, and if that is the case, whether C is a substring for the remaining portion of A, where the match for B ends. Fill in the blanks.

Each blank can have AT MOST one statement. **Marks:** $1+1+2+1+1+1+1+1+1=10$

```
#include <stdio.h>
#include <stdlib.h>
#define MAXLEN 1024
int locateSubstr ( char A[] , char B[] )
{
        int i, j, match;
        if (strlen(B) == 0) return 0;
        for (i=0; i<=__strlen(A) - strlen(B)__; ++i) if (A[i] == B[0]) {
            match = 1;
            for (j=0; j<____strlen(B)____; ++j) if (____A[i+j] != B[j]____)
                { match = 0; break; }
            if (match)_____return i_____;
        }
        return -1;
}
int main ()
{
        char A[MAXLEN], B[MAXLEN], C[MAXLEN];
        /* Assume the code to input strings from the users is here.
           You need not write anything here */
        /* i should store the matching index of B*C in A*/
        int i,j,k;
        if (strlen(B) == 0) i = __locateSubstr(A,C)__;
        else if (strlen(C) == 0) i =  __locateSubstr(A,B)__;
        else{
             j = locateSubstr(A,B);
             if (j < 0) i=j;
             else k = locateSubstr(____&A[j+strlen(B)] , C____);
             if (k>=0) i=_____j_____;
             else i =_____-1_____;
        }
        if (i >= 0)
           printf("The pattern B*C is found in A at idx %d\n", i);
        else
           printf("The pattern B*C is not found in A\n");
        exit(0);
}
```

8. The following recursive function makes the base conversion. It reads two integers n and b from the terminal (with $n \geq 0$ and $b > 1$, both in base 10) and expresses n in base b. For example, the decimal

expansion of 345 in base 10 is $345 = 3 \times 10^2 + 4 \times 10 + 5$. Note that in this case $5 = 345\%10$ and $34 = 345/10$; The output as printed by the program should be: $(345)\_10 = (3,4,5)\_10$. However, please note that for $n = 10$ and $b = 2$, the program should print $(10)\_10 = (1,0,1,0)\_2$ and NOT $(10)\_2 = (1,0,1,0)\_2$.

Fill in the blanks. Each blank can have AT MOST one statement. **Marks:** $2 \times 5 = 10$

```
#include <stdio.h>

void baseconv ( int n , int b )
{
   /* n is too small. Simply print it and return. */
   if (_____n < b_____) { printf("%d",n); return; }

   /* Recursively print the more significant digits */
  _____baseconv(n/b,b)_____;

   /* Finally print the least significant digit */
   printf(",%d",____n%b____);
}

int main ()
{
   int n, b;

   printf("n = "); scanf("%d", &n);
   printf("b = "); scanf("%d", &b);

   if ((n < 0)||(b < 2)) {
      fprintf(stderr, "Error: Invalid input...\n");
      exit(1);
   }

   printf("___(%d)_10 = (____",n);
   baseconv(n,b);
   printf("____)_%d_____",b);

   exit(0);
}
```

9. Answer the following questions. **Marks:** $(4 \times 1) + 2 + (3 \times 2) + (3 \times 2) = 18$

   (a) CIRCLE the correct choice.

   [i] What value will be assigned to the variable a after the following two statements are executed?

```
int a = 7, b = 5, c = -3;
a = a - a % b * c;
```

A. 0	B. 9	C. 13	D. 14

   [ii] What value is assigned to the variable var?

```
#define T 10+10
var = T * T;
```

A. 400	B. 210	C. 200	D. 120

[iii] Which of the following is NOT a legal name of a C variable?

A. 12_pds	B. _12pds	C. pds_12	D. pds12_

[iv] What is the 8-bit 2's-complement representation of -49?

A. 11001110	B. 11001111	C. 10110001	D. 11010001

(b) Find the 32-bit (single-precision) floating point representation of +41.6 in the IEEE 754 format. Put only ONE binary digit in each gap/space provided below.

Sign Bit: 0 **Exponent:** 1 0 0 0 0 1 0 0

Mantissa: 0 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 0 1 1 0

(c) Consider the following for loop:

```
for (k=1; k<100; ++k) k *= k+1;
```

[i] How many times is the statement "$k* = k+1$;" executed? 3

[ii] How many times is the loop condition "$k < 100$" checked in the loop? 4

[iii] What is the value stored in the variable "k" immediately after the for loop terminates? 183

(d) What will be the output of the following programs?

[i]

```
#include<stdio.h>
int main()
{
   int x = 4, y = 6, z = 0;
   while (y != 0) {
      if (y % 2) z += x;
      x *= 2;
      y /= 2;
   }
   printf("%d\n", z);
   return 0;
}
```

Answer: 24

[ii]

```
#include <stdio.h>
int main ()
{
   int a = 5, b = 5, c;
   char p = 'p', q = 'q';
   c = !( (a>=b) || ((a<=b)&&(p>q)) );
   printf("%d\n", c);
   return 0;
}
```

Answer: 0

[iii]

```
#include <stdio.h>
int main ()
```

```
{
   int p, q;
   for (p=q=0; p<10; ++p) {
      q = p + q;
      ++p;
   }
   printf("%d\n", q);
   return 0;
}
```

Answer: 20

10. The following program computes the sum of the square of digits in the decimal representation of a non-negative integer. For example, the sum of the square of digits for 320127 is $3^2+2^2+0^2+1^2+2^2+7^2=67$. Fill in the blanks with appropriate C constructs.

    Each blank can have AT MOST one statement. **Marks:** $1+1+1+1+2+1=7$

```
#include <stdio.h>
int main ()
{
   unsigned int n, d, sum;

   /* Read the unsigned integer n*/
   scanf( _____"%u"_____ , &n );

   /* Initialize sum */
   sum = _____0_____ ;

   /* Loop as long as n is not reduced to zero*/
   while ( ____n > 0 (OR) n != 0____ ) {

      /* Store in d the least significant digit of n*/
      d = _____n%10_____ ;

      /* Add the square of this least significant digit to sum*/
      sum = _____sum + d*d_____ ;

      /* Remove this digit from n */
      n = _____n/10_____ ;
   }

   /* Print the sum of the square of digits of the input integer*/
   printf("The sum of the square of digits is %d \n", sum );

   return 0;
}
```

[Extra Page/ Rough Work]

[Extra Page/ Rough Work]

[Extra Page/ Rough Work]