9

05/10/18

INDIAN INSTITUTE OF TECHNOLOGY ROPAR

Department of Computer Science &Engineering

GE103Introduction to Computing & Data Structures

MidTerm Exam 05Oct 2018

Max. Marks: 40 Time Limit: 120 minutes

Name: [redacted] Roll No: [redacted]

==

NOTE:

- *Read the questions carefully, and write your answers as neatly as possible.*
- *You need to write your answers in the space provided below each question. No extra sheet should be attached to this paper. Rough work may be done in the space provided or in last empty sheet.*
- *Best wishes !*

1. **[2 marks]** Consider a two dimensional array : A[6][8] of total 48 integer elements. If the base address (A) is 1600 and the system uses zero-indexing, what is the memory address of element A[3][4] ? Assume:

(a) Row-major order

1600+ (3×8 +4)×4 = 1600 + 112 = 1712

(b) Column-major order

1600 + (4×6+3)×4 = 1600 + 108 = 1708

2. **[18 marks]** What will be the output for the following codes. Explanation for the output not necessary.

Answers / OUTPUT

```
#include <stdio.h>  /* 2 marks */
int main() {
   float f1; int i=40, j=30, k=20;
   int p=5;
      f1=42/4 + 4.0/3  + 5.24;
      p = i>j>k;
  printf( "f1= %.2f  p=%d", f1,p);
}/* You may use the space here for rough work/calculations */
```

f1 = 17.07

P = 0

Answers / OUTPUT

```
#include<stdio.h>  /* 2.5  marks */
void main() {
    char arr[ ] = {'l', 'a', 't', 'e', 's', 't'};      //First element is L lower case
    char *p = (arr+2);
    printf("%c", *p+2);
    printf("\n %d  %d", sizeof(arr), sizeof(p));
}/* You may use the space here for rough work/calculations */
```

5

6 4

```
#include <stdio.h>  /* 2.5  marks */
void main()
{
 for (int k=1;  k< 4;  )
    printf( "%d \n", ++k );
 }/* You may use the space here for rough work/calculations */
```

2, 3   k=1 → 2
       k=2 → 3
       k=3 → 4.

2 3 4

```
# include <stdio.h>  /* 3 marks */
int main() {
  int i = 0;
  for (i=1; i<20; i++)   {
    switch(i)    {
      case 1:
        i += 1;
      case 2:
        i += 3;
      case 4:
        i += 4;
      default:
        i += 8;
        break;
    }
    printf(" %d ", i);
  }
  return 0;
}/* You may use the space here for rough work/calculations */
```

18+8 = 26
17+8.
25

i = 1+1 → i = 2
i = 2+3 = 5
i = 5+4 = 9

9+8 = 17

i = 17+1 = 18
i = 18+3 =

17   26

Answers / OUTPUT

```
#include <stdio.h>  /* 2 marks */
#define ALPHA 0
#define BETA 1
int main()  {
   int i = 5;
   switch (i & 1)
   {
      default: printf("Default");
      case ALPHA: printf("alpha");        break;
      case BETA: printf("beta");          break;
   }
return 0;
}/* You may use the space here for rough work/calculations */
```

101
001
1

beta

11

```
#include <stdio.h>  /* 3 marks */
int main(){
          int k, sum=0;
          for (k=2048; k ; k >>= 1)
               sum++;
          printf("%d   %o   %x ", sum, sum+1, sum+2);
          return 0;
} /* You may use the space here for rough work/calculations */
```

k = k>>1
2048
sum = 1
(3)
(2)₈
002
2×8°
2×8°
2|2048 - 0
2|1024 - 0
2|512 - 0
2|256 - 0
2|128 - 0
2|64 - 0
2|32 - 0
2|16 - 0
1000000000000
k = 100000000000
k = 1000000000

~~(0002)8 (0003)16~~

1   2   3

```
#include <stdio.h> /* 3 marks */
void main()
{ int i=1, j=5, k=11;
  int *p = &j;   int *q = p;   int *r = &k;
  *p = i;  (*p)++;
   i += 2;
  *r  = *r - *q;
    p=r; j=j+i;
   k =  k+ *q;
   printf( "%d   %d   %d ", i, j, k );
 }/* You may use the space here for rough work/calculations */
```

3   5   11

p = 9
9+2
5 + 11 = 16
p → k
p → j
q = p
q → j
r → k
11 - 2 = 9
*r = 9
i = 1+3
*p = *p + 1
= i+1 = 2
11 - 5 = 6
value p = 1
*p = i
i = i+2
= 2+2
i = 3
p = 2
*p = *p + 1
1+1
j+3
5
p → j
q → p
p = 6 j = 2
*p = 2

3. **[2 marks]** A student wrote following code for reversing an input integer array A of n elements. But on execution, it is observed that the code is wrong. Student approached the TA Raman who replied that there is/are **small mistake(s)** in this code. Spot the **mistake(s)** (**Encircle that line(s)**) & mention what should be the correct statement/expression(s) there.

```
void reverse(int A[ ], int n) {
  int i, j, temp;
  i=0;
  while (i <n) {        i< n/2
       j= n-1-i;
       temp  =  A[ i ];
       A[ i ] = A[ j ];
       A[ j ] = temp;
       i++;
     }
  }
```

4. **[3 marks]** Refer to following partial C code to transpose a square matrix (or say 2D array). Complete the code (.... *part)* without using any additional array and without declaring any additional variable.

```
#include <stdio.h>
#define N 12  /* this value 12 may vary  by program user*/
void main() {
   int A[N][N]; int i,j,k,temp1,temp2;
   printf("\n Input the NxN matrix elements where N= %d . \n", N);
   for (i=0;i<N;i++) {
           for (j=0;j<N;j++)
               scanf("%d ",&(A[i][j]) );
   }
   .... for ( i=0 ; i< N ; i++)
         { for ( j=0; j<N; j++)
                 temp1 = A[i][j];
A[i][j]= A[j][i];   ~~A[i][j] = A[j][j]~~
                    ~~temp2 = A[i][j]~~
                    ~~printf("%d", temp1);~~
                    ~~printf("\n"); }~~
```

```
~~for (i=0; i<N; i++)~~
~~{ for (j=0; j<N; j++)~~
~~temp~~
~~A[i][j] = A[j][i];~~
~~temp2 = A[i][j];~~
~~printf("%d", temp2);~~
~~printf("\n"); }~~
```

```
   printf("\n Following is the TRANSPOSE matrix \n");
   for (i=0;i<N;i++) { printf("\n");
              for (j=0;j<N;j++)
                  printf("%d ", (A[i][j]) );
           }
  }
```

5. **[3 marks]** Consider the following C code that aims to print the multiplication table of input value **n** (assume input n will be positive and less than 100). Will this program give the desired output ? If not, Identify and Remove the errors (Mark / Encircle the wrong statements (if any) and write there correct statements.)

```
#include <stdio.h>
void main() {
   int n,factor,k;
   printf("\n Enter the number for which you need to print multiplication table \n");
   scanf("%d ", n);
   printf("\n Multiplication table is as follows \n");
   factor=1;
   while (factor<=10) {
     k=n * factor;        k = n* factor++ ;
     printf("%d  X  %02d = %d", n, factor, k );
   }
```

13

5. **[5 marks]** Given an input string **inp**, complete the C program below that does the following

- It first computes the total number of those characters that appear twice or more in the input string.
- Then it removes all digits (if any in the input string) and also changes the input string alphabets to lowercase. Then it prints this modified input string as output string.

As an example, if input string **inp** is **"Animesh181SharmAaa"**, the output would be

No. of characters that repeat = 5

Output String: animeshsharmaaa

/*Ans above 5 because A, m, h, 1, and a are the characters that appear again */

```
#include<stdio.h>
/* you are not permitted to use any other library functions */
#define SZ 1000
void main() {
 int i,j,k,temp1,temp2; char c1, c2, c3;
 char inp[SZ];    scanf("%s", inp) ;  temp2 = 0;
 // ....

  ~~for ( i = 0;  [SZ] != '\0';  i++)~~
   ~~{~~      ~~for ( j = i+1;  [SZ] != '\0';  j++)~~
                  ~~if ((~~

   ~~temp1~~
  for (i=0;  inp[i] != '\0';  i++){ temp1 = 0;
    { for ( j= i+1 ;  inp[j] != '\0';  j++)
         if ( inp[i] == inp[j])   temp1 ++;


         if (temp1 = 1). temp2 ++ ;
            printf (" %d ", temp2 ) ;  }

   for (i=0;  inp[i] != '\0';  i++)
    { if ( inp[i]>= 'a' && inp[i] <= 'z')
            { printf(" %c", inp[i]); }
        else if  ( inp[i] >= 'A'  &&  inp[i] <= 'Z')
            {  inp[i] =  inp[i] + 32;
               printf(" %c", inp[i]); }
```

(Note: You may safely assume that size of the input string is less than 1000. You may write the code within the **main** function to achieve the purpose or you may write a separate function e.g. ***int fun1(char *arr)*** and call that function appropriately within **main** function to achieve the purpose)

```
else
        inp[i] = Inp[i]+1 ;
      printf (" %c", inp[i]); }

   }.
```

1¼

19

6. **[7 marks]** Consider a singly linked list (based on NODE structure as mentioned below) referred using the global node pointer variable **head**. Write the C code for successfully deleting the (first appearing) node having data value **key**. If there is no node in the linked list that has data value **key**, the code brings no change to the linked list. If there are multiple nodes with data value **key**, the code deletes that one which appears first while traversing the linked list using global pointer variable **head**.

```
typedef struct node{
        int data;
        struct node * next;
} NODE;
```

Function prototype is as follows - **void find_delete(int key);**

```
struct node * head = NULL ;

void    delete (int key) ;

        struct node *  temp ;

temp = (struct node*) malloc (sizeof (struct node)) ;
        ~~temp->data = key~~

    if ( ~~temp->dat~~ head == NULL)

        head = temp ;
    while ( temp -> next != NULL)
    { if (  temp -> data == key)
            { head -> next = temp -> next ;
              temp -> next = NULL ;
                free (temp) ;
                  temp = NULL ;     }


    else if  ( temp -> data != key)

    temp = temp -> next ;        }.
```