End Term Exam Question Paper
Subject: Computer Programming (CS104)

Section B: Subjective

Important Note: Check that the subjective paper has total 8 pages. Answer all the parts of a question at one place only. Start the answer of every question on a fresh page.

Q1. Analyse the progam below and answer the questions (11 marks)

```
#include<stdio.h>
typedef struct
{
  int day;
  int month;
  int year;
 }date;

 typedef struct person
 {
   char name[10];
   date birthday;
   float salary;
 }emprec;

 void printout(emprec);
 int highest(emprec rec[],int);

 void main( )
 {
   int j;
   emprec record[] = {{"Arun",10,8,75,4000.00} , {"Vinod",3,1,75,3000.00}
 , {"Tanuj",21,1,75,5500.00}};
   ____________________          //function call

   printout(record[j]);
  return;
  }

  int highest(emprec record[],int n)
  {
    float t;
    int i,j=0;
    t=record[0].salary;
    for(i=1;i<n;i++)
    {
       if(t<record[i].salary)
       {
       t=record[i].salary;
       j=i;
       }
    }
    return(j);
  }
```

```
void printout(emprec per)
{
printf("Employee details:\n Name: %s\n Birthdate: %d:%d:%d\n Salary: %f\n",
per.name, per.birthday.day, per.birthday.month, per.birthday.year,
per.salary);
return;
}
```

a) Write the C statement for function call. (1 mark)

b) What is the output of this program? (2 marks)

c) Assume another function **oldest** which determines the oldest person amongst **n** persons. Assume all persons are born in the same year. The function definition is:

```
------ oldest(----------)                                   //Function header
{
int i;
emprec temp=record[0];
for(i=1;i<n;i++)
{
if(temp.____ > record[i]._____)                             //Line 1, compare months
          temp=record[i];
else
   {
    if(_____= =_____)                                       //Line 2; if months are equal
        {if(temp._____>record[i]._____)                     //Line 3,compare days
              temp=record[i];
}
}
}
printf("The oldest person is %s",_____);}                   //Line 4
```

i) Complete function header (1 mark)

ii) Complete Lines 1,2,3,4. (4 marks)

d) Storage for name of person is to be allocated dynamically. The struct **person** is now declared as:

```
typedef struct person
{
Char* name;
date birthday;
float salary;
}emprec;
```

In main, the input for names is now taken as:

```
void main( )
{
emprec record[3];
int i;
char temp[10];
```

```
char temp[10];
char *str;
for(i=0;i<3;i++)
{
printf("Enter name");
scanf("%s",temp);
str=(char*)malloc(_______);      //Line 1
record[i].name=_______;          //Line 2
strcpy(_______); }               //Line 3
```

Complete Lines 1,2,3. (3 marks)

Q2. Analyse the progam below and answer the questions (11 marks)

```
#include <stdio.h>

#define R 4
#define S 3

void fn(float m[R][S]);

int main()
{
      float m[R][S]=
                  {     {35.5,40.5,45.5},
                        {50.5,55.5,60.5},
                        {65.0,70.0,75.5},
                        {80.0,85.0,90.0}
                  };


      fn(m);
      return;
}

void fn(float mm[R][S])
{
      int r,s;
      float t,z;

      for(r=0;r<R;r++)
      {
            t=0;
            for(s=0;s<S;s++)
            {
                  t=t+mm[r][s];
            }
            z=t/S;
            printf("%d \t %f\n",r+1,z);
      }
      return;
}
```

a) What is the output of the program? (3 marks)

b) What will be the output if the array **m** in main function is declared as: (2 marks)

```
float m[R][S]=
                {       {35.5,40.5,45.5},
                        {50.5},
                        {65.0},
                        {80.0}
                };
```

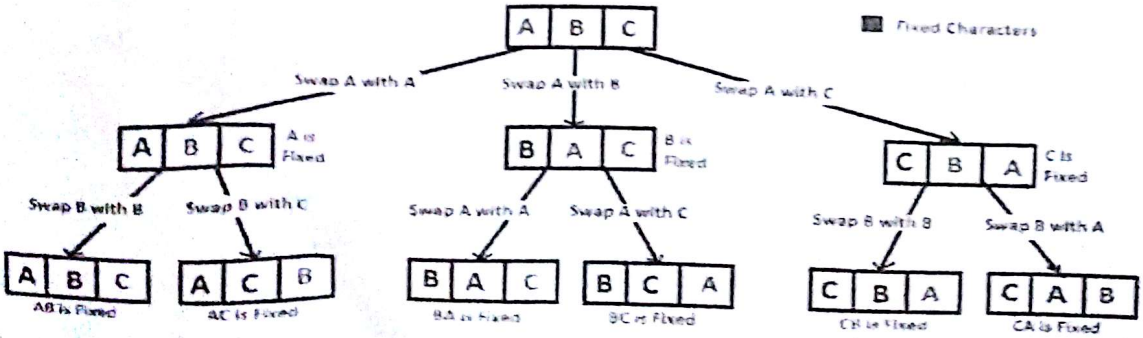
c) A new function **fnnew** prints the highest number in each column of array **m**. The function definition is:

```
void fnnew(float mm[R][S])
{
        int r,s;
        float t;

        for(r=0;-------;r++)            //Line 1
        {
                t=mm[0][r];             //Statement A
                for(s=0;-------;s++)     //Line 2
                {
                        if(t<mm[s][r])          //Line 3
                                ---------;      //Line 4

                }

                printf("%d \t %f\n",r+1,t);     //Statement B
        }
        return;                 //Line 5
}
```

Complete Lines 1,2,4. (3 marks)

d) Assume function **fnnew** now computes the sum of all elements in the array **m** and returns the sum to main. Assume Statements A and B have been commented and sum **t** is initialized to zero. Rewrite the function header and Lines 3,5. (1+1+1 marks)

Q3. Answer questions after studying the below program which prints all permutations of a given string. For example the permutations of string ABC will be ABC, ACB, BAC, BCA, CAB, CBA. Hint: The program implements the logic shown in the given figure.



Recursion Tree for Permutations of String "ABC"

index shor
followir

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

/* Function to swap values at two pointers */
.....

/* Function to print permutations of string
   This function takes three parameters:
   1. a: String array
   2. l: Starting index of the string
   3. r: Index of the last character in the string before '\0' */
void permute(char *a, int l, int r)
{
    int i;
    if (l == r)
       1. .....
    else
    {
        2. for (i = .....; .....; i++)
        {
           swap((a+l), (a+i));  // call to Function swap
          3. .....  // Recursive call
           swap((a+l), (a+i));  // call to Function swap
        }
    }
}

/* Driver program to test above functions */
int main()
{
    char str[] = "ABC";
    int n = strlen(str);
    4.....      //Function Call
    return 0;
}
```

A. Fill in the blanks at lines numbered 1 to 4 and briefly state your reasoning. (2+2+2+1 marks)

B. Write the code of the **swap** function called in function **permute**. (3 marks)

C. What problem you will find in the output of this program, if the characters in the input string are not distinct? (1 mark)

Q4. The primitive root of a prime is defined as follows (refer to the example first): Let p be a prime. A number r is primitive root of p, if first (p -1) powers (0, 1, 2 … p - 2) of r cover all the remainders possible under mod p operation.

For example, primitive root of 7 is 3 because (3^0 mod 7) = 1, (3^1 mod 7) = 3, (3^2 mod 7) = 2, (3^3 mod 7) = 6, (3^4 mod 7) = 4, (3^5 mod 7) = 5. All possible remainders under mod 7 operation have been covered by 6 powers (0, 1, …. 5) of 3.

Given below is a Skeleton program for finding the primitive root of a small prime number. Starting with 2, it tries all values upto the given prime. The first value for which all the remainders are covered is reported as primitive root.

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

1. int checkRemArray(____,____);// check the remainder array for any missing
//remainder from mod operation

int main(){
int prime = 19; // finding the primitive root of 19, say
int root = 2;
for (; root<prime; root++)
{
 int r;// variable for storing value of power
 int remainder[19]={0};// index of the array corresponds to a remainder value
 2. for (r = ____; r<____;r++)
 {
    3. long powerOfRoot = pow(____,____);
    4. int rem= ____ % ____; //find remainder under mod p
    remainder[rem] = 1; //put 1 in the corresponding index of array
  }
  if (checkRemArray(remainder, prime))break;
}
    5. if(____<____)
          printf("The primitive root of %d is %d", prime,root);
    else
          printf("There is no primitive root of %d ", prime);
    return 0;
}
```

A. Fill in the blanks at lines numbered from 1 to 5 to get correct output. (1 x 5 = 5 marks)

B. Write the code for the function `checkRemArray` called in the main. (3 marks)

C. In order to print all the primitive roots of p, write the changes that need to be made in the program. (3 marks)

(a) Let S

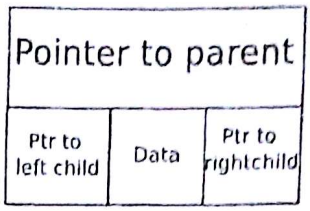
(b) L

2.

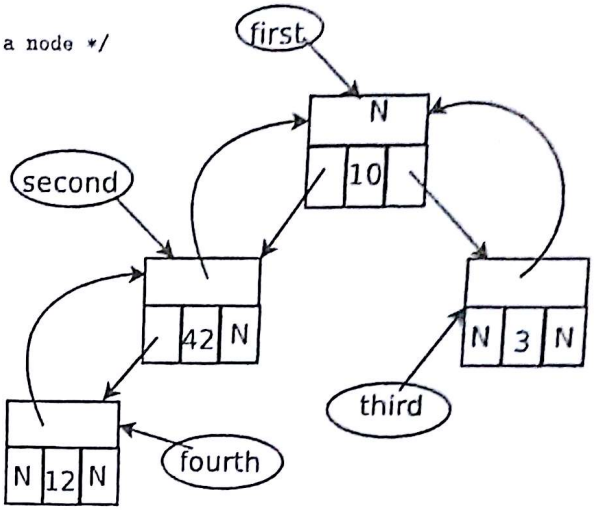
nsider the following program (lets call it ORIGINAL) which implements the shown figure. It compiles without any rror/warning. Answer the questions below. **Hint:** Go through the program and then use figures to understand it.

2+2+1+2+2+3 Marks

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

typedef struct node{
   struct node* parent;              //1
   struct node* left_child;
   struct node* right_child;
   int data;
} NODE;

int main(){
   NODE* first; NODE* second; NODE* third; NODE* fourth;

/* Allocate memory to nodes and make these pointers point to a node */
    first = malloc(sizeof(NODE));
    second = malloc(sizeof(NODE));
    third = malloc(sizeof(NODE));
    fourth = malloc(sizeof(NODE));

 /* Initialize all the four nodes */
    first->data = 10;
    first->parent = NULL;             //2
    first->left_child = second;
    first->right_child = third;

    second->data = 42;
    second->parent = first;           //3
    second->left_child = fourth;
    second->right_child = NULL;

    third->data = 3;
    third->parent = first;            //4
    third->left_child = NULL;
    third->right_child = NULL;

    fourth->data = 12;
    fourth->parent = second;          //5
    fourth->left_child = NULL;
    fourth->right_child = NULL;

    printf("%d %d\n", fourth->parent->data, first->left_child->data);
    /* Statements for (b) and (f) have to be added here. So wherever you write them,
    it is assumed that you meant to write them here. */
  return 0;
  }
```



Structure declaration on the left implements this node. Also note that the size of three pointers in the structure is equal. The diagram is just a convenient way to draw it.



A pictorial representation of the way four nodes are joined. Given this picture and the program you should be able to figure out what do pointers "parent", "left_child" and "right_child" are pointing to. N represents the NULL pointer. The text labels are the names of the pointer to nodes. The arrows originate at a pointer variable and terminate at what they are pointing to. For example, pointer "third" points to a variable of type NODE whose "data" field is 3.

(a) What is the output of the program?

(b) Add C statements in this program (your code should be written between the printf and return statements) to delete node second and let node fourth take its place.

(c) Assume that each node occupies 16 bytes of space in heap memory, i.e., `sizeof(NODE)` returns 16. There are four nodes, so the total space occupied by them is 64 bytes. How much total space is occupied by the nodes in heap memory after the completion of deletion process in step (b)?

(d) If all the lines with the statements that are marked with // (there are five such lines) are deleted from the ORIGINAL program, what will be the output of the new program?

(e) If we write another program without using the pointer parent, i.e, there are only three members in the structure definition. What will be the limitations of the new program as compared to the ORIGINAL?

(f) Add C statements in the ORIGINAL program to insert a node with data value 21 as the right child of the node pointed by third. The added node does not have any left or right child.

Q6. Consider the following program that compiles without any errors but raises a warning stating "`function returns address of local variable`". Answer the questions below.

2+2+2+2+2 Marks

```
#include<stdio.h>
#include<stdlib.h>
int function1 (int x);
int* function2 (int, int* x, int* y);

int main(void)
{
   int i=0; int x=1; int dm=0;
// dm=1;
   x = function1(i);
   int array1[10]={1,1,1};
   int array2[10]={2,3,4};
   int* array3;
   array3=function2(dm,array1,array2);
   printf("%d %d\n", array3[i+2],x);
return 0;
}

int function1 (int x){
   int a,dm;
   int i=42;
   for (a=0;a<10;a++){
       int i = 5;
       i=i+2;
   }
   dm=100;
   return i+1;
}

int* function2 (int dm,int* a, int* b){
   if (dm==0){
      int array[10];
      int i;
      for(i=0;i<10;i++){
         array[i] = a[i] + b[i];
      }
      return array;
   }
   else {
      int* array;
      array=calloc(10,sizeof(int));
      int i;
      for(i=0;i<10;i++){
         array[i] = a[i] + b[i];
      }
      return array;
    }
}
```

(a) What is the output of the program?

(b) In `function2`, in both the `return` statements, variable `array` has been returned. Mr. K thinks that there is a better way of writing that function by writing `return array;` just once, immediately before the last closing } of `function2`. No other statement is added. What will happen if this idea is implemented?

(c) In `function2`, both `if` and `else`, execute the same thing, the two arrays that are taken as parameters are added and put inside a third array. What is the difference between `if` and `else` executions?

(d) What is the output if the comment in the program is removed?

(e) When the program is executing `printf` statement inside `main()`, how many variables are alive, i.e., they have not been destroyed? Justify your answer.