Solution/Model Answer
PCDS-17PCD23 - IAT-2 (2nd sem I & J Section)
Dr. P. N. Singh, Professor(CSE)

1.

a. Write a C Program to check whether a string (with space) is palindrome or not without using built-in functions. 5 M

Ans:

```
/* To check a string(with space) whether it is palindrome
without using built-in function*/

#include <stdio.h>

int main()
{
   char str[50];
   int x,y,len=0,palin=1;
   printf("Enter string : ");
   while((str[len]=getchar())!='\n')
      len++;
   str[len]='\0';   /* last character NULL */
   y=len-1;         /* coming before last character */
   for(x=0;str[x]!='\0';x++,y--)
       if(str[x]!=str[y])
   {
      palin=0;
      break;
   }
   if(palin)
       printf("Yes, \"%s\" is a palindrom\n",str);
   else
       printf("No, \"%s\" is not a palindrome\n",str);

   return (0);
}
```

b. Write a C Program to find GCD of 2 numbers given by user using recursion. 5 M

Ans:

```
/* GCD of 2 numbers using recursive function*/
#include <stdio.h>

int GCD(int dividend,int divider)
{
  int rem=dividend%divider;
  if(rem==0)
   return (divider);
  else
   return GCD(divider, rem);
}

main()
{
   int n1,n2;
```

```
        printf("Enter two numbers : ");
        scanf("%d%d",&n1,&n2);
        printf("GCD = %d\n",GCD(n1,n2));
        return (0);
    }
```

2. Write a C program to find sum and average of marks for 50 students in 6 papers each using 2-d array.
Ans:

```
/*sum and average of marks */
#include <stdio.h>
int main()
{
   int marks[50][6],sum[50],x,y;
   float avg[50];
   for(x=0;x<50;x++)
   {
     sum[x]=0;
     for(y=0;y<6;y++)
     {
       printf("Enter marks for student %d paper %d : ", x+1,y+1);
       scanf("%d",&marks[x][y]);
       sum[x]+=marks[x][y];  /* sum of marks */
     }
     avg[x]=sum[x]/6.00;  /* average of marks for each student*/
   }
    printf("student  p1    p2   p3   p4   p5  p6  sum   average\n");
    for(x=0;x<5;x++)
    {
      printf("%d.",x+1);
      for(y=0;y<3;y++)
          printf("%5d",marks[x][y]);
      printf("%6d%7.2f\n",sum[x],avg[x]);
    }
  return (0);
 }
```

3.

a. Describe string handling functions strlen(), strcpy(), strcat() and strcmp() 5M
Ans:
Header file for all string handling functions is string.h
strlen(str) – returns length of string in number of characters
example:
printf("%d",strlen("PARAS"))
output : 5

strcpy(str1,str2) – Copies str2 to variable str1
example:
char name1[], name2[]="ramaiyah";
strcpy(name1,name2); /* copies name2 to name 1 */

strcat(str1, str2) – Concatenates(appends) str2 ton str1 at the end
example:

char name[]=”Roma”
char surname[]=”Ritambhara”
strcat(name,surname);
printf(“%s”, name);
Output: Roma Ritambhara

strcmp(str1, str2) – Compares two string and returns:
0 if both strings are identical
else returns difference based on ASCII value of first mismatch
printf(“%d”, strcmp(“SINGH”, “SINHA”);
output: - 1

b. Write a C program to check whether a number is prime or not. 5M

Ans:

```
/* to check a number whether it is prime or not */
#include <stdio.h>
#include <math.h>
int main()
{
   int n,d,prime=1;
   printf("Enter number : ");
   scanf("%d",&n);
   for(d=2;d<=sqrt(n);d++)
     if(n%d==0)
     {
   prime=0;
   break;
     }
  if(prime)
        printf("Yes, %d is a prime number\n",n);
  else
        printf("No, %d is not prime number\n",n);
  return (0);
}
```

4. Explain storage class extern, auto, static and register with examples. 10M
Ans:

The storage class specifies the portion of the program within which the variables are recognized. variables can be categorized by storage class as well as data type. Storage class refers to the permanence of a variable and its scope within the program, that is, the portion of the program over which the variable is recognized.

extern (external variables): They are global variables, known to all the functions of the program from point of definition. Hence, they usually span more than one functions, and often an entire program. Variables defined outside of the program are by default of extern storage class.

int global = 200;
int fullmarks = 100; /*external storage class variable */
main( )

{

……

}

auto (Automatic variables): They are local variables, available to the function in which it is declared. Their scope is confined to that function only. Variables declared within a function are by default of auto storage class.

auto int x,y,z;
int num; /* num is also interpreted as auto storage class

static (Static variable): Static variables have the same scope as automatic variables, i.e. they are local to the functions in which they are defined. Unlike automatic variables, however, static variables retain their values throughout the life of the program. As a result, if a function is called again, the static variables defined in the called function will retain their former values **(They will not be reinitialized again. They are confined at the first initialization)**.

```
#include <stdio.h>
int func()
{
   static int k = 0; /* by default also zero */
   k++;
   return (k);
}

int main()
{
     int x, sum=0;
    for(x=1;x<=5;x++)
      sum+=func();
  print("sum = %d\n", sum);
  return (0);
}
```

register (Register variables) : Registers are special storage area within a computer's central processing unit (In-built memory with CPU). The actual arithmetic and logical operations that comprise a program are carried out within this registers. The execution time can be reduced considerably if certain values can be stored within these registers rather than in computer's memory.

The register variables are local to the function in which they are declared. The address of operator cannot '&' cannot be applied to register variables.

register int x, y, z;

5. What is Bubble Sort? Implement Bubble sort for array of integers. 10M

Ans:
Bubble sort: Bubble sort is a sorting techniques to sort the items either in ascending or descending order. In Bubble sort adjacent elements are compared for swapping purpose. 1st element is compared with second, 2nd

with 3rd, 3rd with 4th and so on. In each pass the "heaviest" element sinks down and "lighter" elements "bubble" up.

```
/* program for Bubble sort */
#include <stdio.h>

     int main()
     {
          int num[10],x,y,temp;
          /* Accept values */
          for(x=0;x<10;x++)
          {
               printf("Enter value %d:",x+1);
               scanf("%d",&num[x]);
          }
          /* now sorting */
          for(x=1;x<9;x++)
               for(y=0;y<=10-x;y++)
                    if(num[y] > num[y+1])
                    {
                         temp=num[y];
                         num[y]=num[y+1];
                         num[y+1]=temp;
                    }
          printf("\nSorted value:\n");
          for(x=0;x<10;x++)
               printf("%7d",num[x]);

          return (0);
     }
```

6. Implement Binary Search by a C program for initialized sorted array. 10M

Ans:

```
/* Binary search for sorted array */
#include <stdio.h>
int main()
{
  int a[11]={20,24,26,37,48,59,67,77,88,99,111};
  int x, skey,first,last,mid,found=0;
  printf("The Sorted Array\n");
  for(x=0;x<10;x++)
     printf("%5d", a[x]);
  printf("Enter item to search : ");
  scanf("%d", &skey);
  first=0; last=10;
  while(first <= last && !found)
  {
    mid=(first+last)/2;
    if(a[mid] > skey) last = mid-1;
    else if(a[mid] < skey) first = mid+1;
```

```
        else found=1;
      }
      if(found) printf("Found at %dth position\n", mid+1);
      else printf("Not found\n");
     return (0);
    }
```

7.

a. Define structure. How structure members are accessed using dot (.) operator? 5M

Ans:

Structure is a data structure whose individual elements can differ in type. It may contain integer elements, character elements, pointers, arrays and even other structures can also be included as elements within a structure. struct is keyword to define a structure.

```
struct  tag  { type member1;
               type member2;
               type member3; ...... type member n;};
```

New structure type variables can be declared as follows:

```
  struct tag var1, var2, var3, ....... varn;
```

Example :

```
        struct library {
                    char b_name[30];
                    char author[30];
                    int price;
                 };
```

Now structure variables:

```
        struct library book1, book2, book3;
```

It is allowed to combine both the template declaration and variable declaration in statement:

```
        struct library {
                    char b_name[30];
                    char author[30];
                    int price;
                 } book1, book2, book3;
```

The link between a member and a variable is established using the member operator . (dot operator) or ->

Giving values to members:

```
              book1.price = 76;
```

by scanf function:

```
        scanf (“%d”,&book1.price);
```

b. How structure variables are passed as a parameter to a function? Write with example. 5M

Ans:

Passing structure to a function - There are three methods:

1) To pass each member of the structure as an actual arguments of the function call. It is unmanageable and inefficient when structure size is big.

2) To pass a copy of the entire structure to the called function. All compilers do not support this method. Changes to structure members in called function have no effect to calling function. Again we have it has to return entire copy of the changed structure to the calling function.

3) Address of the structure is passed and structure elements can be accessed indirectly. It is more effective method.

To send a copy of the structure:

```
   function name ( structure variable name )
```

The called function takes the following form

```
   data type function name ( structure type structure name )
             {
                  ..... ;
                  .....;
                  return (expression);
             }
```

When a function returns a structure. It must be assigned to a structure of identical type in the calling function

8. Implement array of structure of students by a C program with fields roll, name and marks and to get data and sort array of structure based on marks in descending order. 5M

Ans:

```
/*sorting array of structure - based on marks in descending order*/
#include <stdio.h>
struct student
{
   int roll;
   char name[20];
   int marks;
}s[1000]; /* array of structure */


 int main()
 {
    int n,x,y;
    struct student temp;
    printf("How many students : ");
    scanf("%d",&n);
    for(x=0;x<n;x++)
    {
      printf("Enter roll, name and marks for student %d : ", x+1);
      scanf("%d%s%d",&s[x].roll,s[x].name,&s[x].marks);
    }
    /*now sorting */
    for(x=0;x<n-1;x++)
      for(y=0;y<n-x-1;y++)
       if(s[y].marks < s[y+1].marks)
       {
        temp=s[y];
        s[y]=s[y+1];
        s[y+1]=temp;
       }
    printf("roll   name    marks:\n");
    for(x=0;x<n;x++)
        printf("%5d%20s%5d\n",s[x].roll,s[x].name,s[x].marks);

    return (0);
}
```

----------------- Think like a person of action; act like a person of thought.------------------