CMR
INSTITUTE OF
TECHNOLOGY

Solution / Model Answer of IAT-2 (May 2017)
Programming in C & data Structure – 15PCD23
II sem – G & F
Dr. P. N. Singh, Professor(CSE)

1. **a) What is array? Explain the declaration and initialization of one and two dimensional arrays with example.** **10**

Ans:
Array is collection of similar data types.
An array is an identifier that refers to the collection of data items which all have the same name. The individual data items are represented by their corresponding array elements. Element ranges from 0 to n-1 where n is total number of elements.

One-dimensional array
Declaration:

syntax: ***datatype arrayname*[size];**
examples:
```
float height[50];
int batch[10];
char name[20];
```

Initialization:
```
int marks[5] = {76,45,67,87,92};
static int count[ ] = {1,2,3,4,5};
static char name[ ] = {'S', 'I', 'N', 'G', 'H'};
```

Two-dimensional array
Declaration:

syntax: type *arrayname*[rowsize][columnsize];
examples:
```
static int stud[16][5];
```

Initialization:
```
int marks[3][3] = {
                    {26,57,66},
                    {56,77,48},
                    {76,54,82}
                  };

static int marks[3][3] = { {0},{0},{0}};
```
Here first element of each row is explicitly initialized to zero while other elements are automatically initialized to zero.

2 a) Write a program to implement the strcpy function. 4

Ans:

strcpy(strvar, strval) function is used to copy string value to string variable. Related header file is string.h

```
/*Program to implement strcpy() function */
#include <stdio.h>
#include <string.h>
int main()
{
   char str1[30], str2[30];
  printf(“Enter string : “);
   scanf(“%s”, str1);
   strcpy(str2,str1);      /* to copy str1 to str2 */
   printf(“copied string is %s\n”,str2);
  return (0);
}
```

2b) Explain any 4 string manipulation functions with examples 6

Ans:

i. strlen( ) : Returns length of string in bytes.

Syntax:

int strlen(str);

Example:

```
printf(“Length of string = %d\n”, strlen(“SINGH”));
```

output will be 5

ii. strcpy( ): copies string

Syntax:

void strcpy(stringvar, stringval);

Example:

```
strcpy(name,”SINGH”);
```

SINGH string is copied to string variable name.

iii. strcat( ): concatenates 2nd string to 1st string

Syntax:

void strcat(name, surname);

Example:

```
strcpy(name,”VIKRAM”);
strcpy(surname,”SINGH”);
```

SINGH will be appended to string variable name and now name is VIKRAMSINGH

iv. strrev( ): reverses string

Syntax:

void strrev(string);

Example:

```
strcpy(name,”VIKRAM”);
strrev(name);
```

Now name is MARKIV

3a) Describe the different methods of passing parameters to functions, with examples. 5

Ans:

Parameters can be passed by two ways:

i. Pass by value/variable
ii. Pass by address

i. Pass by value/variable

When value is passed as parameters to a function its value remains unchanged in calling function.

```
/*pass by value */
#include <stdio.h>
void func(int x)
{
  x=200;
  printf(“value of x in func ( ) =%d\n”,x);  /* prints x = 200 */
}

int main()
{
    int x = 100;
    printf(“value of x in main( ) = %d\n”,x);  /* prints x = 100 */
    func(x);
    printf(“value of x in main( ) = %d\n”,x);  /* prints x = 100 */
    return (0);
}
```

ii. Pass by address

When address of a variable passed as parameters to a function its value may be changed by the called function & changed value will be available in calling function.

```
/*pass by address */
#include <stdio.h>
void func(int *p)  /* p receives address of parameter passed */
{
  *p=200;
  printf(“value in func ( ) changed  =%d\n”, *p);
     /* prints value 200 */
}

int main()
{
    int x = 100;
    printf(“value of x in main( ) = %d\n”,x);  /* prints x = 100 */
    func(&x); /* pass by address */
     printf(“changed value of x in main( ) = %d\n”,x);  /* prints x = 200 */
    return (0);
}
```

3b) Write a C function to count the frequency of vowels and consonants in a string. 5

Ans:

```
/* counts frequency of vowels and total count of consonants */
#include <stdio.h>
#include <string.h>
#include <ctype.h>

void count_vow_conso(char str[ ])
{
   int x, len=strlen(string),a,e,i,o,u,conso;
   char c;
   a=e=i=o=u=conso=0;
  for(x=0;str[x]!='\0';x++)  /* till NULL character occurs */
   {
       if( isalpha(str[x]) )    /* if alphabets from a to z */
       {
            c=tolower(str[x]);   /* converts in lower case */
            if(c=='a') a++;
            else if (c=='e') e++;
            else if(c=='i') i++;
            else if(c=='o') o++;
            else if(c=='u') u++;
            else conso++;
       }
  }
  printf("vowels a=%d e=%d i=%d o=%d u=%d and consonants = %d\n",
                                a,e,i,o,u,conso);
 }
```

4a) What is a structure? Explain the C syntax of structure declaration with an example. How is a structure different from arrays? 5

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.

The individual structure elements are referred to as members.

Structure declaration:

The key word **struct** declares a structure to hold the details of fields. Name of the structure is called structure tag. The tag name may be used subsequently to declare variables that have the tag's element.

```
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;
```

Structure and array :

A structure is collection of dissimilar data types and it represents a records with different types of a field.

An array is collection of similar data type.

If structure is also a derived type then an array is evergreen hero to occupy it. We can make array of structure:

```
          struct library books[10000];
```

4b) Write a program to create of structure to store details of a person and print the details of a person given his name. Assume the fields are Name, Age, Gender and College Name. 5

Ans:

```
/* array of structure */
#include <stdio.h>

struct person
{
  int age;
  char name[20], gender, cname[30];
};
struct person p1;  /* p1 is structure variable */
int main()
{
  printf("Enter name, gender, collegename & age of the person:");
  scanf("%s  %c%s%d",p1.name,&p1.gender,p1.cname,&p1.age);
            /* space before %c */
   printf("Name = %s gender = %c collegename = %s age = %d\n",
                           p1.name,p1.gender,p1.cname,p1.age);
   return (0);
}
```

5a) Explain the concept of array of structures with example. 6

Ans:

An array is collection of similar data type.

If structure is also a derived type then an array is evergreen hero to occupy it. We can make array of structure:

struct tagname structarray[size];

```
struct student
{
  int roll;
  char name[20];
  int marks;
};

struct student s[60];   /* array of structure */

/* to accept rollname and marks of 60 students & to display */

int main()
{
   int x;
   for(x=0;x<60;x++)
    {
       printf("Enter roll, name & marks for student %d : " , x+1);
```

```
        scanf(“%d%s%d”,&s[x].roll,s[x].name,&s[x].marks);
     }

   /* to display */
   for(x=0;x<60;x++)
     printf(“%5d%20s%5d\n”,s[x].roll,s[x].name,s[x].marks);
      /* taking 5 & 20 characters minimum space to display*/
   return (0);
}
```

5b) Write a program to add two matrices. Also check for compatibility.

Ans:

```
/* adding two matrices */
 #include <stdio.h>

 int main()
 {
    int m, n, p,q,c, d, first[100][100], second[100][100],
    sum[100][100];

    printf("Enter the number of rows and columns of matrix1 \n");
    scanf("%d%d", &m, &n);
    printf("Enter the number of rows and columns of matrix2 \n");
    scanf("%d%d", &p, &q);
    if(m!=p || n!=q)  /* checking rows and columns of both matrices
                      for compatibility */
    {
          printf(“Needs square matrix – not compatible\n”);
          return (99); /* return from here with non-zero value*/
    }
    printf("Enter the elements of first matrix:\n");

    for (c = 0; c < m; c++)
       for (d = 0; d < n; d++)
          scanf("%d", &first[c][d]);

    printf("Enter the elements of second matrix:\n");

    for (c = 0; c < m; c++)
       for (d = 0 ; d < n; d++)
             scanf("%d", &second[c][d]);

    printf("Sum of entered matrices:-\n");

    for (c = 0; c < m; c++) {
       for (d = 0 ; d < n; d++) {
          sum[c][d] = first[c][d] + second[c][d];
          printf("%d\t", sum[c][d]);
       }
       printf("\n");
    }

    return (0);
}
```

6a) What is dynamic memory allocation? Explain malloc ( ), calloc ( ) realloc () and free () functions with examples.

Ans:

Dynamically memory allocation:

Memory can be allocated dynamically at run time as & when required. It is good for the fear of wasted or shortage of memory if limited size is declared

For Memory allocation header file is stdlib.h or alloc.h

For allocation we can use malloc( ), calloc( ) and realloc( )
malloc( ) : allocates single block of memory in bytes
example:

```
int *p;
p=malloc(sizeof(int)*20));
```

calloc() : allocates multiple blocks of memory of equal size
calloc(nblocks, size_of_each_block_in_bytes);
example:

```
calloc(10,sizeof(int));
```

realloc() : resets the memory already allocated by malloc() or calloc() by growing or shrinking the size
realloc(nblocks, size_of_each_block_in_bytes);

```
realloc(20,sizeof(int)*2);
```

free() is used to de-allocate the allocated memory generally at the end of execution

example:

```
free(p);
```

memory allocated dynamically should be de-allocated properly at the end of execution

6b) Write a recursive function to find factorial. **3**

Ans:

```
/* recursive function to find factorial */
long int fact(int n)
{
  if(n==0 || n==1)
       return (1);
  else
       return (n*fact(n-1));
}
```

7a) Write a function to find sum, mean, variance and standard deviation using pointers 5

```
/* sum, mean, variance & standard deviation */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main()
{
  double *a,sum=0.00,mean,variance,stdev,diff,sum2;
  int x,n;
  printf("How many numbers : ");
  scanf("%d",&n);
  a=malloc(sizeof(double)*n);  /* allocating memory */
  for(x=0;x<n;x++)
  {
    printf("Enter number %d : ", x+1);
    scanf("%lf",(a+x));
    sum+= *(a+x);       /* using pointers */
  }
  mean=(double)sum/n;
  sum2=0.00;
  for(x=0;x<n;x++)
  {
     diff=*(a+x) - mean;
     sum2+=diff*diff; /* sum of squares of diff. in number & mean */
  }
  variance=(double)sum2/(n-1);
  stdev=sqrt(variance); /* std deviation is square root of variance*/
  printf("Sum=%.3lf Mean=%.3lf Standard deviation=%.3lf\n" ,
                                  sum,mean,stdev);
  return (0);
 }
```

7b) What is a pointer? How do you declare and initialize a pointer? What is a pointer to pointer? Give example 5

"Pointer is a variable that stores address(location) of data as value."
In C when we define a pointer variable we do so by preceding its name with an asterisk.

To declare:

datatype *pointervar;

```
int *p;

/*Example code */
#include <stdio.h>
int main()
{

  int n=20,*p;
  p=&n; /* address of n is initialized to pointer variable p */
  /*Now value of n can be accessed by variable or by address;
       *p value is value and p is address */
  printf("Value of n = %d\n", n);
  printf("Address of n = %u\n", p);
  printf("Value stored at address = %d\n", *p);
         *p=40;
  printf("Value changed by pointer = %d\n", n);
            /* value of n is changed to 40 */
  return (0);
 }
```

Pointer to Pointer : A pointer can store address of another pointer :

Address ---→ Address --→ value

```
        int n, *p, **q;
p=&n;
q=&p;
/* Here p stores address of n and q stores address of p */
```

p	**: address of n**
***p**	**: value of n**
q	**: address of p**
***q**	**: value of p i.e. address of n**
****q**	**:value of n**