C Programming

Year – June 2015

BCA-202

[Time : Three Hours]

[Maximum Marks : 75]

Note : Attempt all the Sections as per instructions.

This solution is provided by **Mrs. Tushina**

Section - A

(Very Short Answer Questions)

Attempt all five questions. Each question carries 3 marks. Very short answer is required not exceeding 75 words.

Q1. What is array in C programming ? Explain with example.

Ans. In C programming Array is a collection of variables belonging to the same data type. Some characteristics of arrays are:

(i) Array might be belonging to any of the data types.

(ii) Array size must be a constant value.

(iii) Contiguous memory locations are used to store array elements in memory.

(iv) It is a best practice to initialize an array to zero or null while declaring, if we don't assign any values to array.

For example:

int a[10] → integer array.

We can refer to the array with the help of pointers as:

```
# include <stdio.h>
  disp (int *num)
   {
     printf("%d", *num);
   }
  void main ( )
   {
     int arr [ ] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 0);
     for (int i=0; i<0, i<=10; i++)
       {
         disp (& arr [i]);
       }
   }
```

Output: 1 2 3 4 5 6 7 8 9 0

Q2. What is a 'Pointer' ? What operators can be used on pointers ?

Ans. Pointers: Pointers are variables which contain address of some other variables.

Declaration:

```
      datatype   *pointername
e.g.  long       *ptr.
```

The type of pointer depends upon the type of the variable it points to. Every pointer points to some datatype.

Arithmetic Operations on Pointers :

1. ***Incrementing pointer:*** Increments its value by the number of bytes of its data type.
2. ***Decrementing pointer:*** Decrements its value by the number of bytes of its data type.
3. ***Pointer comparisons:*** Pointer may be compared by using relational operators such as ==, <and>.
4. Addition/subtration of pointer and number.
5. Differencing between two pointers.

Q3. How can you declare and initialize a string ?

Ans. String is declared as the 'array of characters'.

Syntax: char-variable-name [size];

e.g. char s[10];

String initialization:

1. ***At declaration time:*** The following two methods are used for string initialization.

(a) char name[05] = "GYAN";

(b) char name[05] = {'G', 'Y', 'A', 'N', '\0'};

Here, we have used 05 size for the string because in the last the compiler automatically inserts the '\0' which is called **null character**.

2. *Run time initialization:* Following functions are used for initialization :

(a) scanf ( ):

```
syntax:  scanf ("%s", variable-name);
e.g.     scanf ("%s", str);
```

(b) getchar ( ): used to read only one character.

```
syntax:  variable=getchar ( );
e.g.     char a;
         a = getchar ( );
```

(c) gets ( ): This function is used to get input of a line of string.

```
syntax:  char-variable-name [size];
         gets (variable);
e.g.     char str [20];
         gets (str);
```

Q4. Distinguish between structure and union.

Ans. ***Difference between structure and union:*** Unions and structures in C are same in concepts, except allocating memory for their members.

(i) Structures allocate storage space for all its members seperately, **whereas** Union allocates one common storage space for all its members.

(ii) We can access only one member union at a time, **whereas** in structure, one can access all member values at the same time.

Difference with the help of an example:

```
# include<stdio.h>
union job
{
   char name [32];
   float salary;
   int worker no;
} u;
struct job
{
   char name [32];
   float salary;
   int worker-no;
} s;
void main ( )
{
  printf ("size of union = %d", size of (u));
  printf ("size of structure = %d", size of (s));
}
```

Output :

```
      size of union = 32
      size of structure = 40
```

Q5. Distinguish between #ifdef and #if directive.

Ans. ***Distinguishing between # ifdef and #if directive:***

The #ifdef directive:

The # ifdef directive has the following syntax:

ifdef identifier newline

This directive checks whether the identifier is currently defined. Identifier can be defined by a #define directive or on the command line. If such identifiers have not been subsequently undefined, they are considered currently defined.

The #if directive:

The #if directive has the following syntax:

if constant expression newline

The directive checks whether the constant expression is true (nonzero). The operand must be a constant integer expression that doesn't contain any increment (++), decrement (−−), size of pointer (*), address (&) and cast operators.

Identifiers in the constant expression either are or not macro names.

Section - B

(Short Answer Questions)

Attempt any two questions out of the following three questions. Each question carries 7.5 marks. Short answer is required not exceeding 200 words.

Q6. What are two-dimensional arrays ? How can you initialize them ?

Ans. Two-dimensional array : The simplest form of multidimensional array is 2-D array. A 2-D array is, in essence, a list of 1-D arrays. To declare a 2-D array of size x, y we have to write as :

type arrayname [x] [y];

where **type** can be any valid C-data type
array name → a valid C-identifier

A 2-D array can be thought of as a table which will have 'x' number of rows and 'y' number of columns. A 2-D array 'a', which contains three rows and four columns can be shown as :

	Column 0	Column 1	Column 2	Column 3
Row 0	a[0] [0]	a[0] [1]	a[0] [2]	a[0] [3]
Row 1	a[1] [0]	a[1] [1]	a[1] [2]	a[1] [3]
Row 2	a[2] [0]	a[2] [1]	a[2] [2]	a[2] [3]

Thus every element in array is identified by an element name of the form a[i] [j],
where a → array name
i, j → subscripts that uniquely identify each element in 'a'

Initializing 2-D arrays: Multidimensional arrays may be initialized by specifying bracketed values for each row. Following is an array with 3 rows and each row has 4 columns.

```
int a[3] [4] = {{0, 1, 2, 3.}. {4, 5, 6, 7}. {8, 9, 10, 11}};
```

The nested braces, which indicate the intended row, are optional.

Q7. Write a program in C to concatenate two strings without using strcat( ).

Ans. Program in C to concatenate two strings without using strcat ( ):

```
# include <stdio.h>
void main ( )
  {
    char s1[100], s2[100], i, j;
    printf ("Enter First string:");
    scanf ("%s", s1);
    printf ("Enter Second string:");
    scanf ("%s", s2);
    for (i = 0; s1[i]! = '\0'; ++i);
    for (j = 0; s2[j]! = '\0'; ++j, ++i)
      {
        s1[i] = s2[j];
      }
    s1[i] = '\0';
    printf ("After concatenation: %s", s1);
  }
```

Output:

```
Enter first string; hello
Enter second string: bye
After concatenation: hellobye.
```

Q8. What is 'Union' in C? Explain with an example.

Ans. *Union:* Unions are conceptually similar to structures. All members of union uses a single shared memory location which is equal to the size of its largest data member.

This implies that although a union may contain many members of different types, it cannot handle all the members at the same time. A union is declared using "union" keyword.

e.g.: union item

```
{ int m;
  float x;
  char c;
} t1:
```

This declares a variable t1 of type union item. Here this union contains three members each with a different datatype.

Accessing a union member: To access members of union t1, we can do;

```
t1. m;
t1. x;
t1. c;
```

Complete example of union

```
# include <stdio.h>
# include <conio.h>
union item
  { int a;
    float b;
    char ch;
  }
void main ( )
  { union item t1;
    t1.a = 12;
    t1.b = 20.2;
    t1.ch = 'z';
    clrscr ( );
    printf ("%d\n", t1.a);
    printf ("%f\n", t1.b);
    printf ("%c\n", t1.ch);
    getch ( );
  }
```

Output:

```
-26426
20.19
Z
```