# F.E. Sem - 2 CBCGS SPA – MAY 18

**Q.1.a) Select the correct option from multiple choice questions. (10 M)**

i. **Which bitwise operator is used to multiply the number by $2^n$ where n is number of bits.**
**A]Bitwise-OR B]Bitwise-AND C]Bitwise Left shift D] Bitwise Right shift**
**Ans:- C**

ii. **Which operator has the lowest priority?**
**A] ++ b] % C] + D] ||**
**Ans:-D**

iii. **Which of these is a valid variable declaration?**
**A]in temp salary; B] float marks_student; C] float roll-no; D] int main;**
**Ans:-B**

iv. **What will be the output of the following program?**

```
Void main() {
Double x=28;
Int r;
r=x%5;
printf("\n r=%d",r);}
```

**A] r= 3 B] Run time Error C] Compile time Error D] None of the above**
**Ans:-A**

v. **What will be the output of following program**

```
Void main() {
 int x[]={10,20,30,40,50};
printf("\n%d%d%d%d",x[4], 3[x], x[2], 1[x], x[0]); }
```

**A]Error B}10 20 30 40 50 C] 50 40 30 20 10 D] None of these**
**Ans:-D**

vi. **Which of the following is not a keyword of 'C'?**
**A]auto B]register C]int D]function**
**Ans:-D**

vii. **What will be the output?**

```
Void main() {
Int y;
y=0x10+010+10;
printf("\ny=%x",y); }
```

**A] y=34 B] x=34 C] y=22 D] Error**

**Ans:C**

viii. **Study the following C program**

```
Void main() {
int a=0;
for( ;a;);
a++; }
```

**what will be the value of the variable a, on the execution of above program**

**A] 1 B] 0 C] -1 D] None of these**

**Ans:A**

ix. **Which of the following is used as a string termination character?**

**A] 0 B]\0 C] /0 D]None of these**

**Ans:B**

x.

```
void main() {
char a[]= "Hello world";
char *p;
p=a;
printf("\n%d%d%d%d",sizeof(a),sizeof(p),strlen(a),strlen(p) ); }
```

**A] 11 11 10 10 B] 10 10 10 10 C] 12 12 11 11 D] 12 2 11 11**

**Ans:D**

**Q.1.b) State True or False with reason. (10 M)**

**i. Size of pointer variable is equal to the datatype it points to. True**

**ii. A float constant cannot be used as a case constant in a switch statement. True**

**iii. The statement void p; is valid. True**

**iv. while(0); is an infinite loop. False**

Ans: when the condition is while(0); the control will never enter into the loop hence there will not be any infinite loop.

**v. scanf() function is used to input string having multiple words. True**

**vi. A function can have any number of return statements. True**

**vii. In a union, space is allocated to every member individually. False**
Ans: A union shares space among its member.
**viii. An algorithm is a graphical representation of the logic of a program. False**
Ans: An algorithm is set of structured instruction used to execute the code.
**ix. Comments in the program make debugging of the program easier. True**
**x.There is no difference between '\0' and '0'. False**
Ans: '\0' is used as a string terminator whereas '0' is number for the interpreter.

**Q.2.a.i) How to create array of structure variable and assign values to its members? (5 M)**
**Ans:**
1. An array of structure can be declared in the same way as declaring array of any other data type.
2. For example, an array of the structure student can be declared as shown:
   struct student s[100];
   This array can now store information of 100 students.
3. Array of structure must be used when many variables of a structure are required.
4. Syntax:
```
Struct structure_name
{
    data_type member1;
    data_type member2;
    -
    -
    data_type member;
};
```

**Q.2.a.ii) Differentiate between struct and union. When is union preferred over struct? Give on example of each. (5 M)**

**Ans:**

| Sr. NO. | Structure | Union |
|---|---|---|
| 1. | Memory allotted to structure is equal to the space require collectively by all the members of the structure. | Memory allotted for a union is equal to the space required by the largest memory of that union |
| 2. | Data is more secured in structure. | Data can be corrupted in a union. |
| 3. | Structure provide ease of programming. | Unions are comparatively difficult for programming. |
| 4. | Structures require more memory. | Unions require less memory. |
| 5. | Structure must be used when information of all the member elements of a structure are to be stored. | Unions must be used when only one of the member elements of the union is to be stored. |

1. Union is preferred over struct when only one of the member elements of the union is stored.

   Example of structure:

```
#include<conio.h>
#include<stdio.h>
Struct student
{
    char name[20];
    int roll_no;
    float fees;
};
void main()
{
    Struct student s1;
    clrscr();
    printf("Enter the student's name, roll number ");
    gets(s1.name);
    scanf("%d",&s1.roll_no);
    printf("The student details are as follows: \nName:%s\n
Roll number:%d",s1.name,s1.roll_no);
    getch();
}
```

**Output:**

```
Enter the student's name, roll number:John
20
The student details are as follows:
Name:John
Roll number:20
```

Example of union:

```
#include<stdio.h>
#include<conio.h>
Union info
{
     char name[20];
};
void main()
{
 Union info i1;
int choice;
clrscr();
printf("Enter your name");
scanf("%s",i1.name);
printf("Your name is %s",i1.name);
getch();
}
```

**Output:**

```
Enter your name:John
Your name is John
```

**Q.2.b.i) WAP to print the sum of following series. (5 M)**

**$1+2^2+3^3+\ldots\ldots\ldots+n^n$**

**Ans:**

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int n,sum=0;
 clrscr();
 printf("Enter a number ");
 scanf("%d",&n);
 while(n!=0)
 {
  sum=sum+(n*n);
  n--;
 }
 printf("The sum of the series is %d",sum);
 getch ();
}
```

**Output:**

```
Enter a number 3

The sum of the series is 14
```

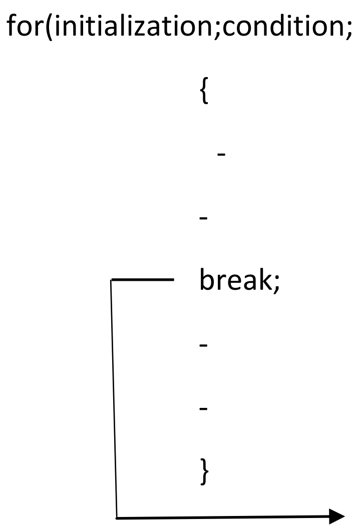
**Q.2.b.ii) Compare the following** **(5 M)**

**i. break and continue statements**

**Ans:**

break statement: The break statement neglects the statement after it in the loop and transfer the control outside the loop.

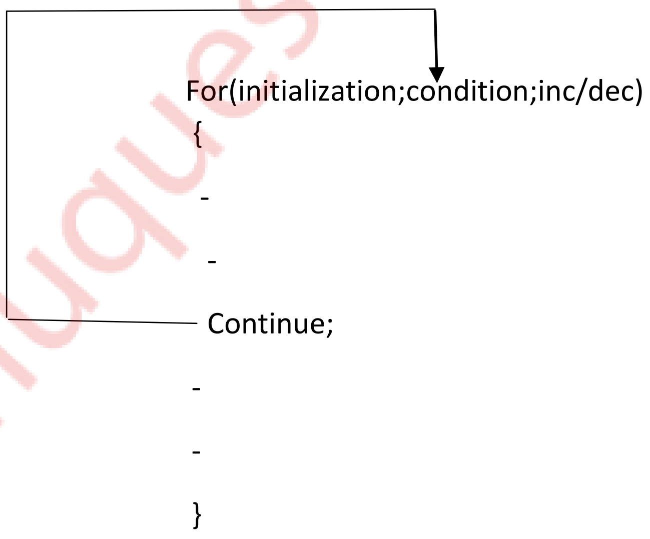
Operation of break statement in for loop:



```
for(initialization;condition;inc/dec)
          {
            -
          -
          break;
          -
          -
          }
```

Continue statement: The continue statement also neglects the statement after it in the loop and transfer the control back to the starting of the loop for next iteration.

Operation of continue statement in for loop:



```
For(initialization;condition;inc/dec)
{
 -
  -
 Continue;
-
-
}
```

**ii. if-else and switch statements:**

**Ans:**

| Sr.no | If……else | Switch |
|---|---|---|
| 1. | In this we can test only one condition. | In this we can test multiple condition. |
| 2. | It can have values based on constraints. | It can have values based on user choice. |
| 3. | In this we cannot have different conditions. | In this case we can have only one expression but various value of the same expression. |
| 4. | If else statement is used to evaluate a condition to be true or false | A switch case statement is used to test for multiple values of the same variable or expression like 1,2,3 etc. |

**Q.3.a) Write a program to calculate number of vowels (a, e, i, o, u) separately in the entered string. (6 M)**

**Ans:**

```
#include <stdio.h>

#include<conio.h>


void main()

{

 int c = 0, count = 0,a=0,i=0,e=0,o=0,u=0,A=0,I=0,E=0,O=0,U=0;

 char s[1000];

 clrscr();


 printf("Input a string\n");

 gets(s);


 while (s[c] != '\0')

 {

  if (s[c] == 'a')

   {
```

```
 a++;
 printf("\na=%d",a);
}
else  if (s[c] == 'A')
{
 A++;
 printf("\nA=%d",A);
}
else  if (s[c] == 'e')
{
 e++;
 printf("\ne=%d",e);
}
else  if (s[c] == 'E')
{
 E++;
 printf("\nE=%d",E);
}
else  if (s[c] == 'i')
{
 i++;
 printf("\ni=%d",i);
}
else  if (s[c] == 'I')
{
 I++;
 printf("\nI=%d",I);
}
else  if (s[c] == 'o')
```

```
       {
        o++;
        printf("\no=%d",o);
       }
       else if (s[c] == 'O')
       {
        O++;
        printf("\nO=%d",O);
       }
       else  if (s[c] == 'u')
       {
        u++;
        printf("\nu=%d",u);
       }
       else  if (s[c] == 'U')
       {
        U++;
        printf("\nU=%d",U);
       }

       count++;
      c++;
    }
    getch();
  }
```

**Output:**

```
Input string University
U=1
i=2
e=1
```

**Q.3.b) Predict output of following program segment.** **(4 M)**

**i.**

```
main()
{
int a,b,*p1,*p2,x,y;
 clrscr();
 a=48;b=10;p1=&a;p2=&b;
 x=*p1**p2-8;
 *p1=*p1+*p2;
 y=(*p1/ *p2)+20;
 printf("%d%d%d%d%d%d",*p1,*p2,a,b,x,y);
}
```

**Output:**

```
5810581047225
```

**ii)**

```
main()
{
 int x=4, y=9,z;
 clrscr();
 z=x++ + --y+y;
 printf("\n%d%d%d",x,y,z);
 z= --x+x+y--;
 printf("\n%d%d%d",x,y,z);
 getch();
}
```

**Output:**

```
5820
4716
```

**Q.3.c) An electronic component vendor supplies three products: transistors, resistors and capacitors. The vendor gives a discount of 10% on order of transistor if the order is more than Rs. 1000. On order of more than Rs. 100 for resistors, a discount of 5% is given and discount of 10% is given on order for capacitors of value more than Rs. 500. Assume numeric code 1, 2 and 3 used for transistors, capacitors and resistors respectively. Write a program that reads the product code and order amount and prints out the net amount that the customer is required to pay after discount. (Note: Use switch-case) (10 M)**

**Ans:**

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int choice,n,dis;
 clrscr();
 printf("\n1.Transistor\n2.Capacitor\n3.Resistor\nEnter your choice");
 scanf("%d",&choice);
 printf("Enter the price");
 scanf("%d",&n);
 switch(choice)
 {
  case 1:if(n>1000)
             {
              dis=n/10;
              dis=n-dis;
              printf("The total price after discount is %d",dis);
             }
             else
             {
              dis=n;
              printf("The total price is %d",dis);
             }
             break;
  case 2:if(n>500)
             {
              dis=n/10;
              dis=n-dis;
              printf("The total price after discount is %d",dis);
             }
             else
             {
              dis=n;
              printf("The total price is %d",dis);
             }
             break;
  case 3:if(n>100)
             {
              dis=n/5;
              dis=n-dis;
              printf("The total price after discount is %d",dis);
             }
             else
             {
```

```
            dis=n;
            printf("The total price is %d",dis);
           }
           break;
   }
  getch();
 }
```

**Output:**

```
1.Transistor
2.Capacitors
3.Resistor
Enter your choice 1
Enter the price 1050
The total price after discount is 945
```

**Q.4.a) What is recursion? WAP using recursion to find the sum of array elements of size n.**

**(10 M)**

**Ans:**

1. Recursion: A function that calls itself is called as recursive function and this technique is called as recursion.
2. A recursive function must definitely have a condition that exits from calling the function again.
3. Hence there must be a condition that calls the function itself if that condition is true.
4. If the condition is false then it will exit from the loop of calling itself again.

Program:

```
#include<conio.h>
#include <stdio.h>
#define MAX_SIZE 100

int sum(int arr[], int start, int len);
```

```
void main()
{
   int arr[MAX_SIZE];
   int N, i, sumof_array;
   clrscr();
   printf("Enter size of the array: ");
   scanf("%d", &N);
   printf("Enter elements in the array: ");
   for(i=0; i<N; i++)
   {
    scanf("%d", &arr[i]);
   }


   sumof_array = sum(arr, 0, N);
   printf("Sum of array elements: %d", sumof_array);

   getch();
}
int sum(int arr[], int start, int len)
{
   if(start >= len)
    return 0;

   return (arr[start] + sum(arr, start + 1, len));
}
```

**Output:**

```
Enter size of the array: 10

Enter elements in the array: 1 2 3 4 5 6 7 8 9 10

Sum of array Elements: 55
```

**Q.4.b) Write a C program to** **(10 M)**

i. **Create a 2D array [Matrix] [in main function]**
ii. **Write a function to read 2D array[Matrix]**
iii. **Write a function that will return true(1) if entered matrix is symmetric or false(0) is not symmetric.**
iv. **Print whether entered matrix is symmetric or not [in main function]**

**Ans:**

Program:

```
#include<stdio.h>

void main()
{
  int m, n, c, d, matrix[10][10], transpose[10][10];
  clrscr();
  printf("Enter the number of rows and columns of matrix\n");
  scanf("%d%d", &m, &n);
  printf("Enter elements of the matrix\n");
  for (c = 0; c < m; c++)
    for (d = 0; d < n; d++)
      scanf("%d", &matrix[c][d]);
  for (c = 0; c < m; c++)
    for (d = 0; d < n; d++)
      transpose[d][c] = matrix[c][d];
  if (m == n)
```

```
 {
   for (c = 0; c < m; c++)
   {
     for (d = 0; d < m; d++)
     {
         if (matrix[c][d] != transpose[c][d])
           break;
     }
     if (d != m)
         break;
   }
   if (c == m)
     printf("The matrix is symmetric.\n");
   else
     printf("The matrix isn't symmetric.\n");
 }
 else
   printf("The matrix isn't symmetric.\n");

 getch();
}
```

**Output:**

```
Enter the number of rows and columns of matrix
2 2
Enter elements of matrix
1 2 3 4
The matrix isn’t symmetric.
```

**Q.5.a) implement string copy function STRCOPY (str1,str2) that copies a string str1 (source) to another string str2 (destination) without using library function. (5 M)**

**Ans:**

```
 #include<stdio.h>
 #include<conio.h>
void main()
{
  char s1[100],s2[100],i;
clrscr();
 printf("enter string s1: ");
scanf("%s",s1);
for(i=0;s1[i]!='\0';++i)
 {
           s2[i]=s1[i];
  }
    s2[i]='\0';
    printf("string s2: %s",s2);
    getch();
}
```

**Output:**

```
Enter string s1:program
String s2: program
```

**Q.5.b) Explain file handling in c in details.[Note: Mention file type, file modes, file related functions and its use) ( 8 M)**

**Ans:**

There are various types of files like: text file stored with the extension "txt", binary files stored with the extension "bin" etc. For file handling or accessing the contents of file, there are certain predefined functions available in the c programming language,

A file represents a sequence of bytes on the disk where a group of Related data is stored. File is created for permanent storage of data. C programming Language can handle files as stream-oriented data (text) files and system oriented Data (binary) files.

An important thing required to access files is the "FILE pointer". This pointer is used to point to the values stored in the file. A file pointer is hence to be created for accessing the files. The syntax for creating a file pointer is a given below:

FILE *<identifier for pointer>;

For e.g. FILE *fp;

Hence in every program we write in this section to access files, we will use this kind of pointer declaration. This pointer is used to point the data to be accessed in the file i.e. whenever a data is read or written in the file, it is from the location pointed by the file pointer "fp".

Function and there use:

1. fopen(): This function is used to open a file to be accessed in the program. The file to be opened is to be passed as a string parameter to the function and also the mode of opening the file is to be passed as the string to the function. Hence the syntax of the function with parameters is as given below:
   <file pointer identifier>=fopen<"file name>",<mode of opening the file>")
   For e.g. fp=fopen("test.txt","w");
   The various modes in which file can be opened are as follows
   1. "r" indicates that the file is to be opened indicates in the read mode.
   2. "w" indicates that the file is to be opened indicates in the write mode.

      When a file is opened in write mode the data written in the file overwrites the previously stored data in the file.

   3. "a" indicates that the file is to be opened in the append mode. In this mode the data written into the file is appended towards the end of the already stored data in the file. The earlier stored data remains as it is.
   4. "w+" indicates that the file is to be opened in write and read mode.
   5. "r+" indicates that the file is to be opened in read and write mode.

2. fclose(): The function is used to close the file opened using the file pointer passed to the function. The syntax with parameters to call this function is as given below:
   fclose(<file pointer identifier>);
   for e.g. fclose(fp);
   this statement closes the file opened using the file pointer variable “fp”. It closes the file opened using the function fopen().
3. feof(): This function returns true or false based on whether the pointer pointing to the file has reached the end of the file or not. The pointer used to point the file has to be passed as a parameter to the function feof(). Also the file has to be opened pointed using the pointer “fp”,before using this function. The syntax of the function with the parameters is as shown below:
   feof(<file pointer identifier>);
4. fputc(): This function is used to put a character type data into the opened file using the open() function, pointed by a file pointer. The character to be put into the file as well as the pointer are to be passed as the parameters to this function. The syntax to call this function along with the parameters to be passed is as shown below:
   fputc<char type data>,<file pointer identifier>);

   for e.g.: fputc(c,fp);

   this example will store the character value of the char type data variable. ”c” into the opened file and pointed by the pointer fp at the position pointed by the pointer fp in the file.

5. getc(): This function is used to get a character from the file pointed by the corresponding file pointer passed to the function. It is exactly opposite the fputc() function. This function brings the character from the file opened and pointed by the file pointer variable passed to the function. The syntax of the function call with the parameters to be passed is as given below:
   getc(<file pointer identifier>);
   For e.g. getc(fp);
   This function brings the character type data from the opened file using the pointer fp; from the location pointed by the pointer “fp” in that file.
6. rewind() : this function is used to rewind or bring the file pointer variable to the point to the beginning of the file from wherever it is currently pointing in the file. The syntax of the function call with the parameters to be passed is as given below:
   rewind(<file pointer identifier>);
   for e.g.rewind(fp);
   this function rewinds or brings back the pointer “fp” to the beginning of the file from wherever it was pointing in the file opened using the pointer “fp”.
7. Fprintf():This function is used to store the different data types in the file as the fputc() function is used to store the character in the file. This can be used to store

integers , float, string etc. types of data into the file opened. The function is similar to printf(),except that it writes to the file instead of the monitor. The syntax shown below explains the concept:
fprintf(<file pointer identifier>,"<format specifiers>",<variable names>);
For e.g.: fprintf(fp,"%d",x);
This function will print or store the value of integer variable "x" in the file opened using the pointer "fp". The data will be stored at the location pointed by the pointer variable "fp".

8. Fscanf(): The function is used to read the different types of the data as the getc() Function is used to read a character from the file. This function can be used to read an integer, float string etc. types of data into the file opened. The function is similar to scanf(),except that it reads from the file instead of the keyboard. The syntax shown below explains the concept:
fscanf(<file pointer variable>,"<format specifiers>",<address of the variables in which the data is to be read>);
for e.g.: fscanf(fp,"%d",&x);

**Q.5.c) WAP to print all possible combination of 1,2,3 using nested loops.(7 M)**

Ans:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i,j,k;
clrscr();
for(i=1;i<=3;i++)
        {
                for(j=1;j<=3;j++)
                        {
                                for(k=1;k<=3;k++)
                                printf("\n%d%d%d",i,j,k);
                        }
        }
getch();
```

```
        }
```

**Output:**

```
111
112
113
121
122
123
131
132
133
211
212
213
221
222
223
.
.
.
.
  (and so on.)
```

**Q.6.a) WAP to print following pattern for n lines. [Note: range of n is 1-9] (5 M)**

```
      1
     121
    12321
   1234321
```

Ans:

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int i,j,n;
        clrscr();
        printf("Enter the number of lines:");
        scanf("%d",&n);
        for(i=1;i<=n;i++)
        {
                for(j=1;j<=n-i;j++)
                {
                        printf(" ");
                }
                for(j=1;j<=i;j++)
                {
                        printf("%d",j);
                }
                for(j=i-1;j>=1;j--)
                {
                        printf("%d",j);
                }
                printf("\n");
        }
        getch();
}
```

**Output:**

```
           1
          121
         12321
        1234321
```

**Q.6.b) WAP to print binary equivalent of entered decimal no. (5 M)**

Ans:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
{
        int n,i;
        clrscr();
        printf("Enter a number: ");
        scanf("%d",&n);
        printf("Binary form is: ");
        for(i=15;i>=0;i--)
        {
                printf("%d",n/(int)(pow(2,i)));
                n=n%(int)(pow(2,i));
        }
        getch();
}
```

**Output:**

```
                Enter a number:12
                Binary form is:0000000000001100
```

**Q.6.c) what is significance of storage classes? Explain it with relevant examples. (10 M)**

Ans:

The different locations in the computer where we can store data and their accessibility, initial values etc. very based on the way they are declared. These different ways are termed as different storage classes.

In C there are for storage classes, namely

1. Automatic
2. Register
3. Static
4. External or global

Let us see these storage classes one by one

1. Automatic storage class
   In this case data is stored in memory
   The initial value of such a variable is garbage
   The scope of the variable is local i.e. limited to the function in which it is defined.
   The life of such variables is till the control remains in the particular function where it is defined.
   For e.g.:
   Int i; or auto int i;

2. Register storage class
   In this case data is stored in CPU register
   The initial value of such a variable is garbage.
   The scope of the variable is local i.e. limited to the function in which it is defined

The life of such variables is till the control remains in the particular function where it is defined.
For e.g.:
Register int I;
In this case the data is stored in a small memory inside the processor called its registers.
The advantage of such storage class is that since the data is in the processor itself, its access and operation on such data is faster.
There is limitation on the size of the data that can declared to be register storage class. The data should be such that it doesn't require more than 4 bytes. Hence double and long double data types cannot be declared as a register.
Also there is a limitation on the maximum number of variables in a function that can be a register class. The limitation is that a maximum of 3 register class variable can be declared in a function.

3. Static storage class
   In this case data is stored in a memory
   The initial value of such a variable is zero
   The scope of the variable is local i.e. limited to the function in which it is defined
   The life of such variable is till the program is alive.
   For e.g.:
   Static int I;
   If a variable is declared static, its value remains unchanged even If the function execution is completed.
   When the execution to that function returns, the previous value is retained.
   Thus it says the initialization is only once. If you have an initialization statement of a static member, it will be executed only once i.e. for the first time when this function is called.
4. External or global storage class
   In this case data is stored in memory
   The initial value of such a variable is zero.
   The scope of the variable is global i.e. it is accessible from anywhere in the program.
   The life such a variable is till the program is alive.