F.E. Sem - 2 CBCGS SPA – MAY 17

Q.1.a.) Convert 0010 0100 0010 1101 from base 2 to decimal. Convert 134 from base 10 to hexadecimal. Write steps for conversion.

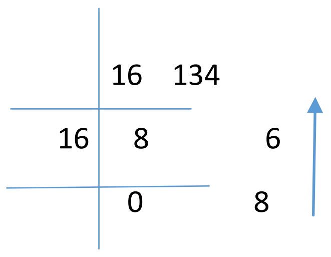
Ans:

Binary to decimal: The steps to be followed are:

1. Multiply the binary digits (bits) with powers of 2 according to their positional weight.
2. The position of the first bit (going from right to left) is 0 and it keeps on incrementing as you go towards left for every bit.
3. Given: $(0010\ 0100\ 0010\ 1101)_2 = (?)_{10}$
4. $0*2^{15}+0*2^{14}+1*2^{13}+0*2^{12}+0*2^{11}+1*2^{10}+0*2^{9}+0*2^{8}+0*2^{7}+0*2^{6}+1*2^{5}+0*2^{4}+1*2^{3}+1*2^{2}+0*2^{1}+1*2^{0}$
   =0+0+8192+0+0+1024+0+0+0+0+32+0+8+4+1
   =9261
   $(0010\ 0100\ 0010\ 1101)_2 = (9261)_{10}$

Decimal to hexadecimal: The steps to be followed are:

1. Divide the decimal number by 16. Treat the division as an integer division.
2. Write down the remainder (in hexadecimal).
3. Divide the result again by 16. Treat the division as an integer division.
4. Repeat step 2 and 3 until result is 0.
5. The hexadecimal value is digit sequence of the reminders from last to first.
6. Given: $(134)_{10} = (?)_{16}$



$(134)_{10} = (86)_{16}$

Q.1.b) Enlist all data types in C language along with their memory requirement, format specifiers and range.

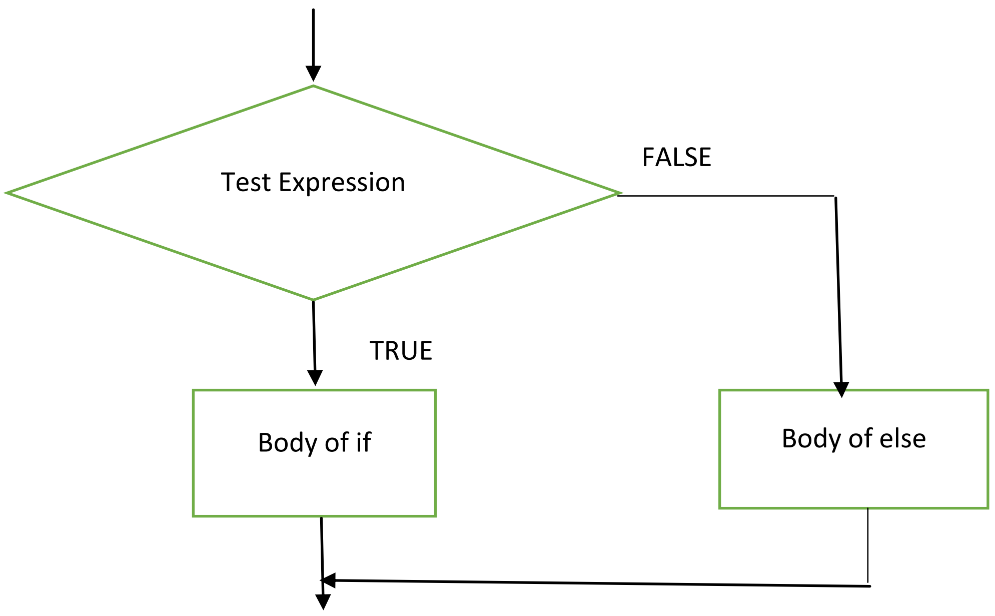
Ans:

Sr. NO.	Data Type	Type of data to be stored	Range	Space required in memory (bytes)
1.	Char	1 character in ASCII form	-127 to 128	1
2.	**Signed char**	**1 character in ASCII**	**-127 to 128**	**1**
3.	**Unsigned char**	**1 character in ASCII form**	**0 to 255**	**1**
4.	**Int**	**Integer nos.**	**-32768 to 32767**	**2**
5.	**Signed Int**	**Integer nos.**	**-32768 to 32767**	**2**
6.	**Unsigned Int**	**Integer nos.**	**0 to 65535**	**2**
7.	**Short Int**	**Integer nos.**	**-32768 to 32767**	**2**
8.	**Long int**	**Integer nos.**	**-2147483648 to 2147483647**	**4**
9.	**Signed short int**	**Integer nos.**	**-32768 to 32767**	**2**
10.	**Signed Long int**	**Integer nos.**	**-2147483648 to 2147483647**	**4**
11.	**Unsigned short int**	**Integer nos.**	**0 to 65535**	**2**
12.	**Unsigned long int**	**Integer nos.**	**0 to 4294967296**	**4**
13.	**Float**	**Fraction nos.**	**$3.4e-^{38}$ to $3.4e^{38}$**	**4**
14.	**Double**	**Fraction nos.**	**$1.7e^{-308}$ to $1.7e^{308}$**	**8**
15.	**Long double**	**Fraction nos.**	**$3.4e^{-4932}$ to $1.1e^{4932}$**	**10**

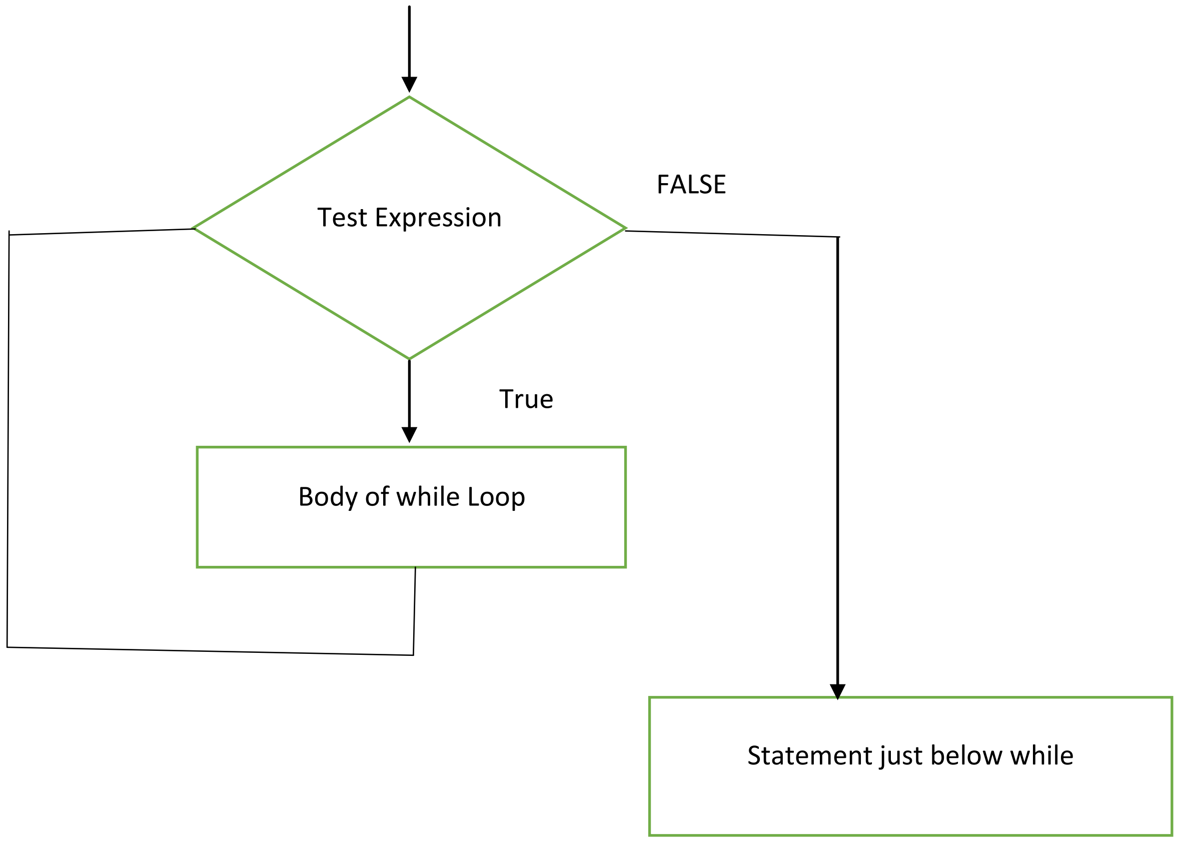
Q.1.c) Draw flowchart for “if…else” and “while” statement of C language.

Ans:

If…..else flowchart



While flowchart



Q.1.d) What is the usage of storage classes? Explain extern storage classes with suitable example.

Ans:

1. The different locations I the computer where we can store data and their accessibility, initial values etc. vary based on the way they are declared. These different ways are termed as different storage classes.
2. In C, we have four storage classes namely
   i. Automatic
   ii. Register
   iii. Static
   iv. Extern or Global.

Extern Classes:

1. In this case data is stored in memory.
2. The initial variable of such variable is zero.
3. The scope of the variable is zero i.e. it is accessible from anywhere in the program.
4. The life of such variable is till the program is alive.
5. Let us see the example

```
#include<stdio.h>
#include<conio.h>
int a=5;
void main
{
 Int a=10;
 printf("%d\n",a);
 printf("%d\n",::a);
a=::a+a;
printf("%d\n",a);
printf("%d\n",::a);
::a=a;
printf("%d\n",a);
printf("%d\n",::a);
getch();
}
```

Output:

```
10

5

15

5

15

15
```

Q.1.e) Differentiate between structure and union.

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.

Q.2.a) What is the need of computer programming. What do you mean by structured programming? Develop an ALGORITHM and FLOWCHART to find reverse of a number.

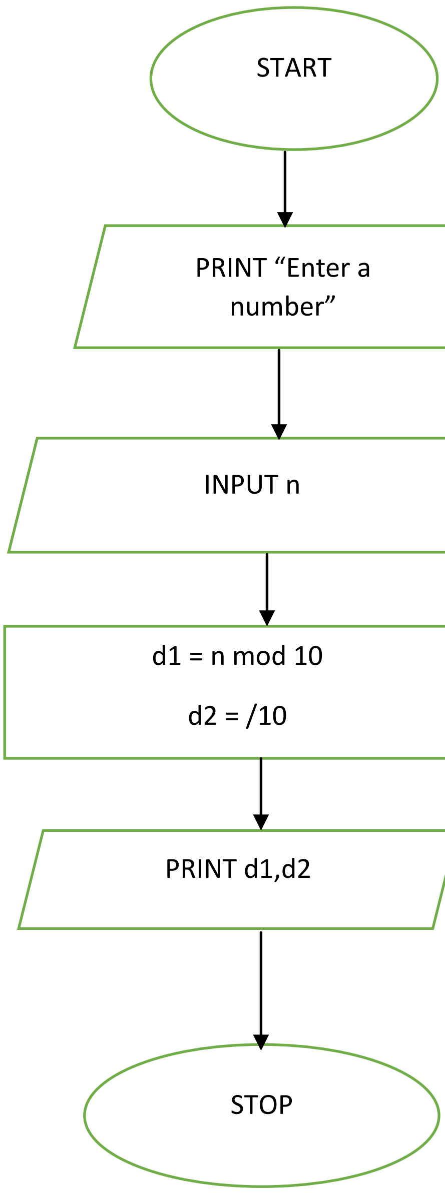
Ans:

1. Programming is to give the machine a list of steps to perform a particular task.
2. If the system to which programming is done is a computer than it is called as Computer programming.
3. A programming of any system has to be done in the language understood by that system.
4. Programming languages are an interface between humans and computer, they allow us to communicate with computers in order to do awesome things.
5. Structured programming (SP) is a technique devised to improve the reliability and clarity of programs
6. In SP, control of program flow is restricted to three structures, sequence, IF THEN ELSE, and DO WHILE, or to a structure derivable from a combination of the basic three. Thus, a structured program does not need to use GO TOs or branches (unless it is written in a language that does not have statement forms corresponding to the SP structures, in which case, GO TOs may be used to simulate the structures).
7. The result is a program built of modules that are highly independent of each other.
8. In turn, this allows a programmer to be more confident that the code contains fewer logic errors and will be easier to debug and change in the future.
9. However, SP may be less efficient than an unstructured counterpart.

Algorithm:

```
1. START
2. PRINT “Enter a number”
3. INPUT n.
4. d1 = n mod 10
5. d2 = n/10
6. PRINT d1, d2.
7. STOP.
```

Flowchart:


START
PRINT “Enter a number”
INPUT n
d1 = n mod 10
d2 = /10
PRINT d1,d2
STOP

Q.2.b) Write a menu driven program to perform arithmetic operations on two integers. The menu should be repeated until user selects "STOP" option. Program should have independent user defined functions for each case.

Ans:

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int operator,ans;
double firstNumber,secondNumber;
clrscr();
 do
{
printf("Enter a operation:\n1.Addition\n2.Subtraction\n3.Multiplication\n4.Divison");
scanf("%d", &operator);
printf("Enter two operands: ");
scanf("%lf %lf",&firstNumber, &secondNumber);
switch(operator)
{
  case 1:
      printf("%.1lf + %.1lf = %.1lf",firstNumber, secondNumber, firstNumber +
secondNumber);
      break;
    case 2:
      printf("%.1lf - %.1lf = %.1lf",firstNumber, secondNumber, firstNumber -
secondNumber);
      break;
  case 3:
      printf("%.1lf * %.1lf = %.1lf",firstNumber, secondNumber, firstNumber *
secondNumber);
      break;
      case 4:

         printf("%.1lf / %.1lf = %.1lf",firstNumber, secondNumber, firstNumber /
secondNumber);

         break;

      default:

         printf("Error! operator is not correct");

   }

   printf("\nDo you want to continue 1.YES 2. NO");

   scanf("%d",&ans);

   }

   while(ans==1);

   getch();

}
```

Output:

```
Enter an Operation:

1.  Addition
2.  Subtraction
3.  Multiplication
4.  Division
    1
    Enter two operands 10 20
    10.0 + 20.0 = 30.0
    Do you want to continue 1.YES 2.NO 1

    Enter an operation

    1.  Addition
    2.  Subtraction
    3.  Multiplication
    4.  Division
        2
        Enter two operands: 50 60
        50.0 – 60.0 = -10.0
        Do you want to continue 1.YES 2.NO 2
```

Q.3.a) Write a program to create two integer of size 8 and 7. Initialize the arrays with random values. Sort the arrays in ascending order with the help of user defined function namely "sort array". Merge these arrays with the help of another user defined function named "merge arrays" which returns a new array. Program should display the arrays before and after sorting, also the merged arrays.

Ans:

```
#include<conio.h>

#include<stdio.h>

void main()

{

    int a[25],b[25],sum[50],i,j,k=1,n,m,s,temp;

    clrscr();

    printf("Enter the number of element in first array :");

    scanf("%d",&n);

    printf("\nEnter the element of array :\n");

    for(i=1;i<=n;i++)

    scanf("%d",&a[i]);

    printf("\nEnter the number of element in second array :");

    scanf("%d",&m);

    printf("\nEnter the element of array :\n");

    for(i=1;i<=m;i++)

    scanf("%d",&b[i]);

    s=m+n;

    for(i=1;i<=s;i++)

    {

            if(i<=n)

            {

                    sum[i]=a[i];

            }

            else
```

```
                {
                        sum[i]=b[k];
                        k=k+1;
                }
        }
        printf("\n Array before sorting is\n");
        for(i=1;i<=s;i++)
        printf("%d\t",sum[i]);
        for(i=1;i<=s;i++)
        {
                for(j=1;j<=s;j++)
                {
                        if(sum[i]<=sum[j])
                        {
                                temp=sum[i];
                                sum[i]=sum[j];
                                sum[j]=temp;
                        }
                }
        }

        printf("\nElement of array after sorting is :\n");
        for(i=1;i<=s;i++)
        printf("%d\t",sum[i]);
        getch();
    }
```

Output:

```
Enter the number of elements in first array: 8

Enter the element of array:

1 4 7 8 9 22 66 11

Enter the number of elements in second array: 7

33 55 88 14 16 18 79

Array before sorting is

1 4 7 8 9 22 66 11 33 55 88 14 16 18 79

Array after sorting is

 1 4 7 8 9 11 14 16 18 22 33 55 66 79 88
```

Q.3.b) What are advantages and disadvantages of recursion? Comment on conditions to be considered while writing a recursive function. Write a program to print Fibonacci series up to N terms using a recursive function.

Ans:

Recursion: A function that calls itself is called as recursive function and this technique is called as recursion.
Advantages:
1. Reduce unnecessary calling of functions.
2. Through Recursion one can solve problems in easy way while its iterative solution is very big and complex.
3. Extremely useful when applying the same solution.

Disadvantages:

1. Recursive solution is always logical and it is very difficult to trace.

2. In recursive we must have an if statement somewhere to force the function to return without the recursive call being executed, otherwise the function will never return.

3. Recursion uses more processor time.

i. A recursive function must definitely have a condition that exits from calling a function again.

ii. Hence there must be condition that calls the function itselfif that condition is true.

iii. If the condition is false then it will exit from the loop of calling itself again.

Program:

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int n,i;
 int fibo(int,int,int);
 clrscr();
 printf("Enter the number of elements");
 scanf("%d",&n);
 printf("0\n");
 for(i=1;i<=n-1;i++)
 {
   printf("%d\n",fibo(0,1,i));
 }
 getch();
}
int fibo(int a, int b int i)
{
 if(i==1)
 return b;
 else
 return(fibo(b,a+b,--i));
}
```

Output:

```
Enter the number of elements:10
0
1
1
2
3
5
8
13
21
34
```

Q.4.a) What are structures? Comment on nested structures. Write a program to read Title, Author and price of 10 books using array of structures. Display the records in ascending order of Price.

Ans:

Structure:

1. Structure is a collection of multiple data elements that can be of different data types.

2. Array is a collection of data items of same type, while structure can have data items of different types.

3. The memory space required to store one variable of structure is equal to the memory space required by all the elements independently to be stored in memory.

4. syntax

```
struct structure_name

{

  data_type variable_name;

  data_type variable_name;

  -

  -

}
```

Program:

```
#include <stdio.h>


struct book

{

   int price;

   char title[80];

   char author[80];

};


void accept(struct book list[80], int s);

void display(struct book list[80], int s);
void bsortDesc(struct book list[80], int s);

void main()
{

    struct book data[20];
    int n;
    clrscr();

    accept(data,10);
    bsortDesc(data, 10);
    display(data, 10);
    getch();
}

void accept(struct book list[80], int s)
{
    int i;
    for (i = 0; i < s; i++)
    {

            printf("\nEnter title : ");
            scanf("%s", &list[i].title);

            printf("Enter Author name: ");
            scanf("%s",&list[i].author);
            printf("Enter price : ");
            scanf("%d", &list[i].price);
    }
```

```
}

void display(struct book list[80], int s)
{
   int i;

   printf("\n\nTitle\t\tAuthor\t\tprice\n");
   printf("-------------------------------------\n");
   for (i = 0; i < s; i++)
   {
          printf("%s\t\t%s\t\t%d\n", list[i].title, list[i].author, list[i].price);
   }
}

void bsortDesc(struct book list[80], int s)
{
   int i, j;
   struct book temp;

   for (i = 0; i < s ; i++)
   {
          for (j = 0; j < (s -i); j++)
          {
             if (list[j].price > list[j + 1].price)
             {
                    temp = list[j];
                    list[j] = list[j + 1];
                    list[j + 1] = temp;
             }
```

```
            }
    }
}
```

Output:

```
Enter Title: physics
Enter Author name: Einstein
Enter Price:1000

Enter Title:english
Enter Author name:willey
Enter Price:900

Enter Title:chemistry
Enter Author name:narayan
Enter Price:800

Enter Title:history
Enter Author name:george
Enter Price:700

Enter Title:geography
Enter Author name:bailey
Enter Price:600

Enter Title:AOA
Enter Author name:Koreman
Enter Price:500
Enter Title:SPA
Enter Author name:Harish
Enter Price:400

Enter Title:Maths
Enter Author name:kumbhojkar
Enter Price:300

Enter Title:CG
Enter Author name:hern
Enter Price:200

Enter Title:python
Enter Author name:naryan
Enter Price:100

 Title            Author           price
________________________________________________________________
Python           Narayan          100
CG               hern             200
Maths            kumbhojkar       300
SPA              Harish           400
AOA              koreman          500
Geography        bailey           600
History          George           700
Chemistry        Narayan          800
English          willey           900
Physics          Einstein         1000
```

Q.4.b. i) Explain gets() and puts() functions of C language. Comment on their parameters and return values.

Ans:

gets():

1. gets() function is used to scan a line of text from a standard input device.
2. This function will be terminated by a newline character.
3. The newline character won't be included as part of the string. The string may include white space characters.

   Syntax :

   char *gets(char *s);

4. This function is declared in the header file stdio.h.
5. It takes a single argument. The argument must be a data item representing a string. On successful completion, shall return a pointer to string s.

puts():

The C library function int puts(const char *str) writes a string to stdout up to but not including the null character. A newline character is appended to the output.

Example: int puts(char const*str)

Parameters: str-this is the C string to be written

Return value: If successful, non-negative value is returned. On error, the function returns EOF.

Q.4.b. ii) Enlist bitwise operator in c language. Explain any 2 with examples.

Ans.

The bitwise operators are listed below

1. ~ to perform bitwise NOT operation.
2. & to perform bitwise AND operation.
3. | to perform bitwise OR operation.
4. ^ to perform bitwise EXOR operation.
5. << to perform bitwise left shift operation.
6. >> to perform bitwise right shift operation.

AND operator
Example .

5 & 3 = 1

$(5)_{10} = (0\ 1\ 0\ 1)_2$
$(3)_{10} = (0\ 0\ 1\ 1)_2$

$(0\ 0\ 0\ 1)_2 = (1)_{10}$

OR operator

Example.

12 | 9 = 13

$(12)_{10} = (1\ 1\ 0\ 0)_2$

$(9)_{10} = (1\ 0\ 0\ 1)_2$

$(1\ 1\ 0\ 1)_2 = (2)_{10}$

Q.5.a) Write a programs to print following patterns for N lines.

1.
```
5 4 3 2 *
5 4 3 * 1
5 4 * 2 1
5 * 3 2 1
* 4 3 2 1
```

Program:

```
#include<stdio.h>
#include<conio.h>
void main()
{
   int i,j;
   clrscr();
   for(i=1;i<=5;i++)
      {
      for(j=5;j>=1;j--)
          {
           if(i==j)
             printf("*");
            else
              printf("%d",j);
            }
        printf("\n");
         }
     getch();
    }
```

Output:

```
5 4 3 2 *
5 4 3 * 1
 5 4 * 2 1
 5 * 3 2 1
  * 4 3 2 1
```

Q.5.b) 2.

```
        5
       5 4
      5 4 3
     5 4 3 2
    5 4 3 2 1
```

Program:

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int i,j;
  clrscr();
   for(i=5;i>=1;i--)
    {
        for(j=1;j<i;j++)
          printf(" ");
        for(j=5;j>=i;j--)
              printf("%d",j);
            printf("\n");
       }
        getch();
}
```

Output:

```
.         5
         5  4
        5 4 3
       5 4 3 2
      5 4 3 2 1
```

Q.5.b) Why files are needed? Explain all the opening modes. Write a program to create copy of the user specify names of source and destination files.

Ans. Need of files:

1. When a program is terminated, the entire data is lost. Storing in a file will preserve your data even if the program terminates.

2. If you have to enter a large number of data, it will take a lot of time to enter them all. However if you have a file containing all the data, you can easily access the contents of the file using few commands in c.

3. You can easily move your data from one computer to another without any changes.
4. The various modes in which a file can be opened are as listed below:

i. "r" indicates that the file is to be opened indicates in the read mode.

ii. "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.

iii. "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.

iv. "w+" indicates that the file is to be opened in write and read mode.

v. "r+" indicates that the file is to be opened in read and write mode.

```
//program to  create a copy of file.

    #include <stdio.h>
```

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

void main()
{
   char ch, source_file[20], target_file[20];
   FILE *source, *target;
  clrscr();
   printf("Enter name of file to copy\n");
   gets(source_file);

   source = fopen(source_file, "r");

   if( source == NULL )
   {
      printf("Press any key to exit...\n");
      exit(EXIT_FAILURE);
   }

   printf("Enter name of target file\n");
   gets(target_file);

   target = fopen(target_file, "w");

   if( target == NULL )
   {
      fclose(source);
      printf("Press any key to exit...\n");
      exit(EXIT_FAILURE);
  }

  while( ( ch = fgetc(source) ) != EOF )
    fputc(ch, target);

  printf("File copied successfully.\n");

  fclose(source);
  fclose(target);

  getch();
}
```

```
Output:
Enter name of file to copy
Factorial.c
Enter name of targt file
Factorial-copy.c
File copied successfully
```

Q.6.a) Comment on dynamic memory allocation. Write a program to read and store N integers in a Array, where value of N is defined by user. Find minimum and maximum members from the Array.

Ans. Dynamic memory allocation:

1. Dynamic Memory Allocation refers to managing system memory at runtime.
2. Dynamic memory management in C programming language is performed via a group four functions namely malloc(), calloc().
3. These four dynamic memory allocation function of the C programming language are defined in the C standard library header file <stdlib.h>. Dynamic memory allocation uses the heap space of the system memory.

malloc()

1. malloc() is used to allocate a block of memory on the heap.
2. Specifically, malloc() allocates the user a specified number of bytes but does not initialize.
3. Once allocated, the program accesses this block of memory via a pointer that malloc() returns.
4. The default pointer returned by malloc() is of the type void but can be cast into a pointer of any data type.
5. However, if the space is insufficient for the amount of memory requested by malloc().
6. Then the allocation fails and a NULL pointer is returned.
7. The function takes a single argument, which is the size of the memory chunk to be allocated.

```
//program to find maimum and minimum element from the array

  #include<stdio.h>

  #include<conio.h>

  #define MAX_SIZE 100

  void main()

 {

   int arr[MAX_SIZE];

   int i, max, min, size;

  clrscr();

  printf("Enter size of the array: ");

   scanf("%d", &size);

  printf("Enter elements in the array: ");

  for(i=0; i<size; i++)

  {

    scanf("%d", &arr[i]);

   }

 max = arr[0];

 min = arr[0];

 for(i=1; i<size; i++)

 {
```

```
                 if(arr[i] > max)
                {
                 max = arr[i];
                 }
                if(arr[i] < min)
                {
                 min = arr[i];
                }
            }
              printf("Maximum element = %d\n", max);
             printf("Minimum element = %d", min);
             getch();
}
```

Output:

```
Enter size of the array: 10
Enter elements in the array:
6
4
3
8
9
12
65
87
54
24
Maximum element=87
Minimum element=3
```

Q.6.b) Explain any 4 functions from string.h header file with suitable examples.

Ans:

Functions from string.h header file are as follows

<u>strlen() function</u>

1. This function returns an integer value that is the length of the string passed to the function.
2. When returning the length of the string it does not consider the space required for null Character.
3. Hence it returns the exact length of the string neglecting these space required for null character.

Example:

```
#include<conio.h>
#include <stdio.h>
#include<string.h>
{
  Int l;
 char a[100];
 clrscr();
 printf("enter a string\n");
   gets(a);
  l=strlen(a);
  printf("the length of the entered string is: %d",l);
   getch();
 }
```

```
Output:
        Enter a string
        Hello
        The length of the entered string is 5
```

2. strcpy() function

This function copies the second string into the first string passed to the function. The second string remains unchanged. Only the first string is changed and gets a copy of the second string.for e.g. strcpy(str1,str2), will copy the string “str2” into the string “str1”.

Example:

```
#include<conio.h>
#include<stdio.h>
#include<string.h>
void main()
{
  char a[100],b[100];
   clrscr();
   printf(“enter a string\n”);
   gets(a);
  strcpy(b,a);
  printf(“the new string is %s”,b);
 getch();
 }
```

Output:

```
Enter a string
Hello, how are you?
The new string is Hello, how are you?
```

3.Strcmp() function

This function compares the two string variables passed to it.it returns an integer value equal to:

1. 0(zero),if the two strings are equal.
2. Negative value ,if the first string is smaller than the second string.
3. Positive value,if the first string is greater than the second string.

Both the strings remain unchanged.

The string is smaller means its alphabetical sequence is smaller. For

Example: ”Hello” is lesser than “Hi”; because the first character “H” is same, but the second character “e” is smaller than “I”. ”e” is smaller than “I” because “e” comes before “i” in the alphabets i.e. A,B,C, ……Z. the function compares the ASCII value of the characters.

Example:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
    char    a[100],b[100];
    clrscr();
    printf(“enter two strings:\n”);
    gets(a);
    gets(b);
    if(strcmp(a,b)==0)
        printf(“strings are equal “);
    else
        printf(“%s string is greater “,a);
    else
        printf(“%s string is greater”,b);
    getch();
}
```

Output:

```
Enter two strings:

Hello

Hi

Hi string is greater
```

4. strcat() function

This function concatenates (joins) the two string variables passed to it. It returns a string of the combination of the two in the first string variable.

Example:

```
#include<conio.h>

#include<stdio.h>

#include<string.h>

void main()

{

   char a[100],b[100];

   clrscr();

   printf("enter two strings:\n");

   gets(a);

   gets(b);

   strcat(a,b);

   printf("the concatenated strig is %s",a);

   getch();

}
```

Output:

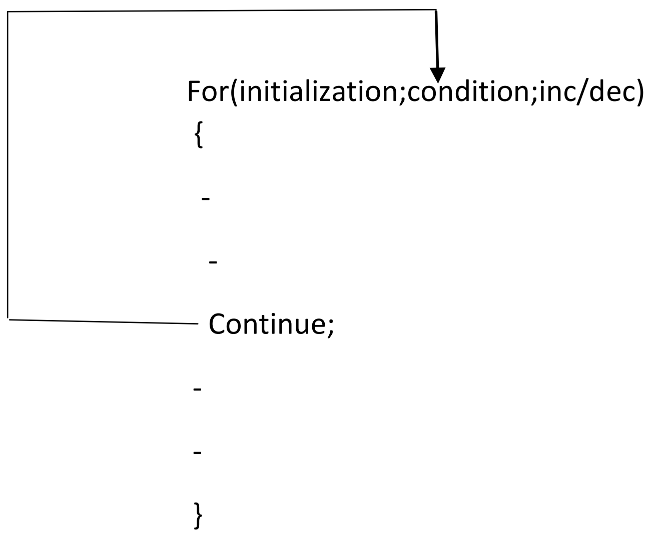
```
Enter two strings:

Mumbai

University

The concatenated string is MumbaiUniversity
```

Q.6.c) Explain continue and break statements with the help of suitable examples.

Ans.

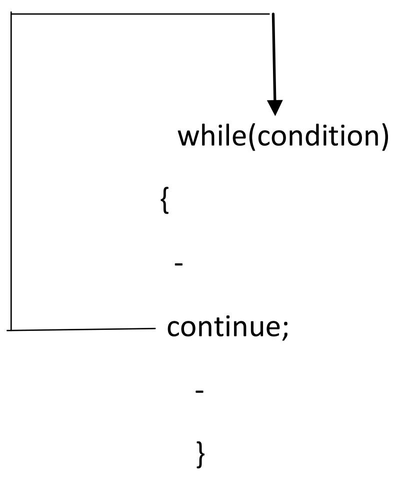
1. continue statement-the continue statement also neglects the statements after it in the Loop and transfers the control back to the starting of the loop for next iteration.

***operation of continue statement in a for loop.**



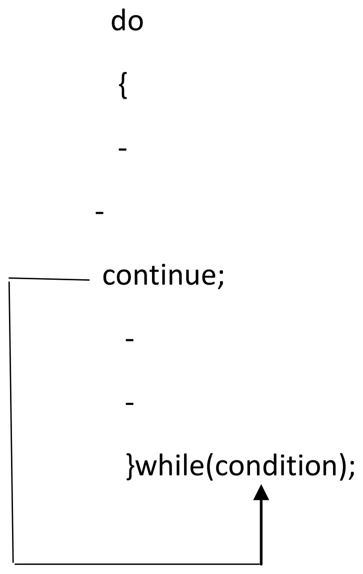
```
For(initialization;condition;inc/dec)
{
 -
  -
  Continue;
-
-
}
```

***operation of continue statement in a while loop**



```
          while(condition)
        {
         -
       continue;
           -
           }
```

***operation of continue statements in a do-while loop**

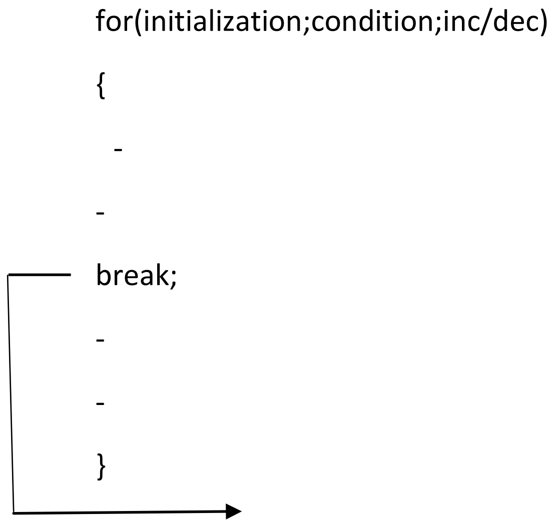


```
   do
   {
   -
  -
  continue;
    -
    -
    }while(condition);
```

2.Break statement:

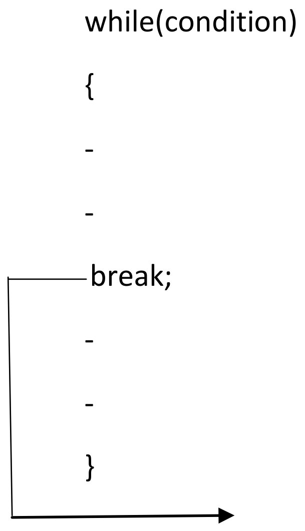
The break statement neglects the statement after it in the loop and transfers the control outside the loop

***operation of break statement in a for loop**



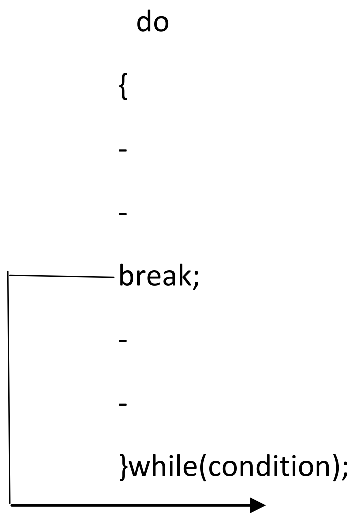
```
for(initialization;condition;inc/dec)
{
  -
-
break;
-
-
}
```

***operation of break statement in a while loop**



```
while(condition)
{
-
-
break;
-
-
}
```

***operation of break statement in a do-while loop**



```
  do
{
-
-
break;
-
-
}while(condition);
```