PAGE NO.:
DATE: / /

KIST.

Group. A.

Q.1) Flowchart is the graphical representation of the sequence of the process of the computer programming. It helps to understand ~~more easily~~ the program flow more easily.

Q.2) Pre-processor directive are the statements which begin with the symbol # (hash) and do not require a ; (semicolon) at the end. They are processed before compilation of any source code in the program.

Q.3) The function getch() and getche() reads a single character. The difference between them is that getch() reads the charcter typed without echoing it on the screen while getche() reads the character and echoes (displays) it into the screen.

Q.4) Ternary operator is an operator represented by '?' together with ':'. It requires three operands.
This is written as:

exp1 ? exp2 : exp3.

where: exp1 is condition. exp2 & exp3 are value or expression.

PAGE NO.:
DATE: / /

Q.5.→ we can declare and initialize array in C programming in following way:-

data-type name[size]; // Declaration
eg: int a[5].

data-type name[size] = {value1, value2, ---};
// initialization.
eg: int a[5] = {1, 2, 3, 4, 5}.

Q.6.→ A function prototype is simply the declaration of a function. It is used to specify the function name, parameter & return. It is also used to give information to compiler that the function may be used later in program.

Q.7 → The difference between do-while and while loop are given below:

S.N.	do-while	while loop.
1.	The do-while loop is exist controlled loop i.e the body of the loop is executed	The while loop is entry controlled loop i.e test condition is evaluated first & body of loop

PAGE NO.:
DATE: / /

	first without checking condition and at the end of body of the loop, the condition is evaluated.	is executed only if this is true.
2.	The body of loop is always executed at least once.	The body of loop may not be ~~always~~ executed.
3.	syntax: do { statement body; } while (condition);	syntax:- while (condition) { statement body; }.

Q.8:> Type casting (Explicit type conversion) means to convert a value from one type to another type in a situation where the compiler ~~do~~ will not do it automatically.

we can achieve it ~~by~~ in this way:

(datatype) expression;

eg: int a;

(float) a; // a will be converted into float type

PAGE NO.:
DATE: / /

Q.9 → Puts() is a function which is used to display the string on the screen. It can't be formatted to print any other data type. It's syntax is.

puts(string_variable);

While, printf() is a function that can print different data types like int, char, float, etc. ~~It~~ It's syntax is.

~~put~~ printf("control string", arg1, ar2, —argn);

Q.10 → The switch is a multiple brach selection statement that successively tests the value of an expression against a list of integer or constant.

The syntax of swith case is :-

```
switch (expression)
{
  case constant1:
    { statements;
      break; }
  case constant2:
    { statements;
      break; }
  default:
      statements;
}.
```

Kist

Q. 11. A program to reverse a number.

```
# include <stdio.h>
# include <conio.h>
int main( )
{
  int n, r=0, rev=0;
  printf("Enter a number to reverse");
  scanf("%d", &n);
  while(n>0)
  {
   r= n % 10;
   rev = rev*10 + r;
   n=n/10;
   }
  printf("The reverse of a number is %d", rev);
  getch();
  return 0;
  }.
```

PAGE NO.:
DATE: / /

Q.12 write a program to check if no is odd or even.

```
#include <stdio.h>
#include <conio.h>
int main()
{
int n;
printf("Enter a number to check odd or even");
scanf("%d",&n);
if (n % 2 == 0)
{
printf("%d is even number", n);
}
else
{
printf("%d is odd number", n);
}
getch();
return 0;
}
```

Q.13. Write a program to print sum of squares of numbers upto n (where n is taken as input from user). if 3 then output is 14 i.e $1^2+2^2+3^2$.

```
#include <stdio.h>
#include <conio.h>
int main()
{
int n, i=1, s=0;
printf("Enter a number");
scanf("%d",&n);
while (i<=n)
{s = s + i*i;
 i++; // It work faster than i=i+1 becoze it uses only one operands
}
printf("The sum of squares of numbers upto %d is %d", n, s);
getch();
return 0;
}.
```

PAGE NO.:
DATE: / /

Q14. write a program to read line of text & print the number of vowels in it.

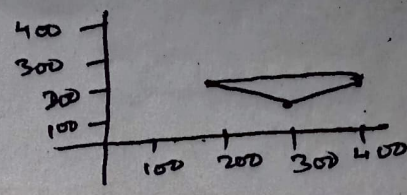
```
#include<stdio.h>
#include<conio.h>
#include <string.h> #include<ctype.h>
int main()
{
  char v[50];
  int i=0, vowel=0;
  printf("Enter a line of text");
  gets(v);
  strlwr(v);
  printf("The vowel letters is");
  while(str[i]!='\0')
  {
  if(str[i]=='a'||str[i]=='e'||str[i]==
  'o'||str[i]=='i'||str[i]=='u')
  {printf("%c\t",str[i]);
   vowel++;
   }
   i++;
  }
  printf("The number of vowel is %d",vowel);
  getch();
  return 0;
}
```

PAGE NO.:
DATE: / /

Q.15. write a program to convert the string to lower case.

```
#include<stdio.h>.
# include<conio.h>
~~#include<ctype.h>~~ # include<string.h>.
int main()
{
char s[100];
printf("enter a string");
gets(s);
~~printf("The lower case of string is %s"~~
strlwr(s)
printf("The lower case of string is %s", s);
getch();
return 0;
}.
```

(300, 100)



PAGE NO.:
DATE: / /

Q.16. write a program to print the triangle using graphics.

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void main()
{
int gd = DETECT, gm;
initgraph(&gd, &gm, "C:\\tc\\bgi");
line(300, 100, 200, 200);
line(300, 100, 400, 200);
line(200, 200, 400, 200);
getch();
closegraph();
}
```

7. Bitwise Operators

Bitwise Operators are used for manipulating data at bit level. These operators are used for testing the bits, or shifting them right or left. Bitwise operators can be applied only to integer type signed or unsigned) and not to float or double. The bitwise operators are in C are as follows:

Operator	Meaning
&	Bitwise AND
|	Bitwise OR
^	Bitwise exclusive OR
<<	Left shift
>>	Right shift
~	Bitwise one"s complement operator

Bitwise Complement (~) is the unary operator and produces a value with each bit of the operand inverted. For example: 1101 0011 becomes 0010 1100.

Bitwise AND (&) is AND and produces a value in which each bit is set to 1 only if both corresponding bits in the two operands are 1.

For example:

```
        1011 0010      (178)
&       0011 1111      (63)
=       0011 0010      (50)
```

Bitwise OR(|) is OR and produces a value in which each bit is set to 1 if either, or both, corresponding bits of the two operands are 1.

For Example :

```
        1011 0010      (178)
|       0000 1000      (63)
=       1011 1010      (186)
```

Bitwise exclusive OR (^) is EXCLUSIVE OR and produces a value in which each bit is set to 1 only if one or the other (but not both) of the corresponding bits of the two operands is 1. For Example :

```
        1011 0010      (178)
^       0011 1100      (63)
=       1000 1110      (142)
```

Example 10: Program that demonstrates the use of bitwise logical operators.

```
#include <stdio.h>
int main()
{
        int num1 = 505;
        int num2 = 215;
        int AND, OR, XOR;
        AND = num1 & num2;
        OR = num1 | num2;
        XOR = num1 ^ num2;
        printf("AND=> %d\n", AND);
        printf("OR=> %d\n", OR);
        printf("XOR=> %d", XOR);
return 0;
}
Output:
AND=> 209
OR=> 511
XOR=> 302
```

PAGE NO.:
DATE: / /

Q.18 WAP in with function swap() to swap the two integer variable using pointers as function parameter.

```
#include<stdio.h>
#include<conio.h>
int swap(int *p, int *q);
int main()
{
int a=5, b=10;
int *p, *q;
p=&a;
q=&b;
clrscr();
printf("Before swap : %d \t %d", a, b);
swap(p, q);
printf("\n After swap : %d \t %d", *p, *q);
getch();
return 0;
}
int swap(int *p, int *q)
{
int c;
c = *p;
*p = *q;
*q = c;
```

```
return *p, *q;
}
```

PAGE NO.:
DATE: / /

Q.19.

a) The process of allocating and freeing memory at run time is called Dynamic memory Allocation in C. This reserves the memory required by the program and returns this resource to the system once the use of reserved space utilized.

We can ~~m~~ allocate ~~and~~ or de-allocate the memory space during Run time (time of execution). C provides a collection of in-built functions for using / managing dynamic memory allocation.

For. eg: To allocate the user required memory space for an integer type dat.

```
~~#include<stdio.h>~~
include n;
int *p;
printf("Enter Number of Elements");
scanf("%d", &n);
p=(int*) malloc(n * sizeof(int));

//~~eg~~ To free (deallocate) the memory
void free(p); // this function will free
              // the memory.
```

Date :.....
Page :.....

Q20. WAP to copy the content of file1.txt to file2.txt.

```
#include<stdio.h>

int main()
{
FILE *fp1, *fp2;
char ch;
fp1 = fopen("file1.txt", "r");
fp2 = fopen("file2.txt", "w");
if (fp1 == NULL && fp2 == NULL)
{
printf("The file opening is not successful");
else
{
ch = ~~get~~ getc(fp1); // read first character;
& moves file pointer next.
while (ch != EOF)
{
fputc(ch, fp2);
ch = getc(fp1);
}
}
return 0;
}
```

Q.2 I→ A structure is a collection of variables referenced under one name, providing a convenient means of keeping related information together. It is heterogeneous collection of related data elements that share a common name.

The general form of structure declaration is :-

```
struct   name
keyword  identifier(user defined)
{
// list of variable.
// declaration (can be of different data type)
};
```

eg:

```
struct book
{
int ISBN;
char title[25];
float price;
};
```

members of struct of different data types.

Group-D

21 WAP to create a structure called person with data members name, age and address. Then createn an array of person type to hold the record of 10 persons. Finally display the information of all the persons whose age is greater than 45.

```
#include<stdio.h>
#include<conio.h>
struct person
{
 char name[50];
 int age;
 ~~add~~ char Add[50];
} p[10];
int main()
{
 int i;
 for(i=0;i<10;i++)
 { fflush(stdin);
 printf("Enter your name");
 ~~gets(name)~~;
 gets(p[i].name);
```

PAGE NO.:
DATE: / /

```
printf("Enter your age");
scanf("%d", &p[i].age); fflush(stdin);
printf("Enter your Address");
gets(p[i].add);
}
for(i=0; i<10; i++)
{
if(p[i].age>45)
{
printf("\nname Name : %s"); p[i].name);
puts(p[i].nam
printf("\nAd : %d", a
printf("\nAge : %d", age p[i].age),

puts(p[i].add);
}
getch();
return 0;
}
```

PAGE NO.:
DATE: / /

Q.22

→ The programming construct that contains control statements to ~~construct~~ control the flow of execution in a program is called control structure. It combines instruction into logical unit. control structure enables us to specify the flow of program control.

The switch is a multiple branch selection statement that successively tests the value of an expression against a list of integer or character constant. When a match is found, the statements associated with that ~~control~~ ~~are~~ constant are executed.

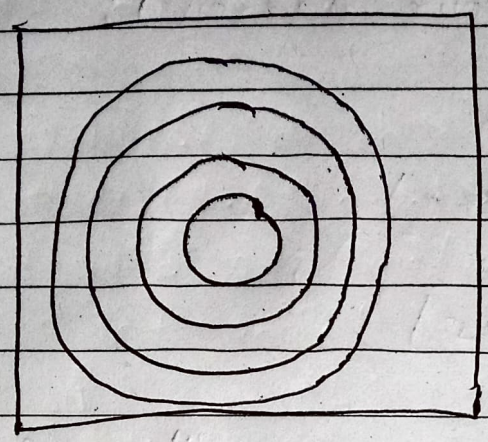
It is written as :-

```
switch(expression)
{
    Case constant1:
        statements;
        break;
    Case constant2:
        statements;
        break;
    default:
        statements;
}.
```

main()

PAGE NO.:
DATE: / /

Q.22. write a program to display the following output using graphics.h library in C.



```
#include<stdio.h>.
# include<conio.h>.
#include<graphics.h>
 int main( )
 {
  int gd=DETECT, gm;
  initgraph(&gd, &gm, "C:\\TurboC3\\BGI"
  circle (200, 500, 60);
  circle (200, 500, 50);
  circle (200, 500, 40);
  circle (200, 500, 20); // A/c to question we require
                         3 circles only..
  outtextxy(500, 0, "CIRCLE");  only to look use
                                or to know use.
  closegraph();
   return 0;
  }.
```