Ashwani kr. Chaudhary
ACE077BCT020.

Q.(i)

The differentiate betn entry controlled & exit controlled loop are:

Entry Controlled	Exit Controlled
(i) Entry control loop checks condition first & then body of the loop will be exected.	(i) The exit controll loop first executes the body of the loop & checks condition at last.
(ii) the body of the loop may or may not be executed at all.	(ii) The body of the loop will be executed at least once because the condition is checked at last.
(iii) for, while are an example of an entry control loop	(iii) Do-while is an example of exit control loop

Q.N.(i)

Program

```
# include <stdio.h>
# include <conio.h>
void main ()
{
    int min, hcf, lcm;
    int num, num1, i, count=0, count1=0;
    printf ("Enter number: ");
    scanf ("%d%d", &num, &num1);
    for (i=1; i<=num; i++)
```

```
{
  if (num % i == 0)
  {
    count++;
  }
  for (i=1; i<=num1; i++)
  {
    if (num1 % i == 0)
    {
      count1++;
    }
  }
if (count != 2 && count1 != 2)
{
 min = (num<num1) ? num : num1;
while (min!=0)
{
 if (num % min == 0 && num1 % min == 0)
{
  break;
 }
else
  {
    min--;
  }
}
 hcf = min;
 lcm = (num * num1) / hcf;
```

```
        printf("HCF is %d \n Lcm is %d", hcf, lcm);
    }

    else
    {
        printf ("prime");
    }
    getch ();
}
```

Q.N.2

⇒ A recursive function is a function that calls itself during its execution. The process may repeat several times, outputting the result & the end of each iteration.

Q.N.3

Program

```
#include <stdio.h>
#include <conio.h>
int sum series (int n)
{
  if (n==1)
  returned 1;
  if (n==2)
  return (sum series *n) *10 + sum series (n-1);
}
void main ()
    {
      int n, result;
      printf ("Enter value of n");
      scanf ("%d", &n);
      result = sum series (n);
      printf ("the sum = %d", result);
      getch ();
    }.
```

Q.N. (3)

⇒ call by value

In call by value, the value of the argument are passed to the function. (The actual value is not passed but the copied value is passed to the function.

eg

```
#include <stdio.h>
int sum (int a, int b);
void main()
{
    int a=5, b=5.5;
    S = sum (a,b);
    printf ("sum is %d", S);
    getch ();
}
int sum (int a, int b)
{
    return (a+b);
}
```

The

In the way call by value is performed in C.

Call by reference

Here the address of the variable is passed to the function as an argument.

Since the address is passed to the function the function can modify the value of variable in calling function using * & &

eg:

```
#include <stdio.h>
void swap (int *a, int *b);
void main ()
{
  int a=5; b=2;
  swap (&a, &b);
  printf ("a=%d, b=%d", a, b);
  getch();
}
void swap (int *a, int *b)
{
  int temp;
  temp = *a;
  *a = *b;
  *b = temp;
}
```

The function modifies the actual value

Program.

```
#include <stdio.h>
#include <conio.h>
void Rev (int T[], int t);
void Read (int T[], int t);
void Display (int T[], int t);
void main()
{
    int Arr[100], i, N, temp;
    printf("How many element,\n");
    Scanf("%d", &N);
    printf("Enter %d element \n", N);
    Read (Arr, N);
    Rev (Arr, N)
    printf("Array in reverse is \n");
    Display (Arr, N);
    getch();
}
void Rev(int Arr[100], int N)
{
    int i, temp;
    for (i=0; i<(N/2); i=i+1)
    {
        temp = *(Arr+i);
        *(Arr+i) = *(Arr+N-i-1);
        *(Arr+N-i-1) = temp;
    }
}
```

```
void Read (int Arr[100], int N)
{
   int i;
   for (i=0 ; i<N ; i=i+1)
   {
      scanf ("%d ", (Arr+i));
   }
}

void Display (int Arr[100], int N)
{
   int i;
   for (i=0 ; i<N ; i=i+1)
   {
      printf ("%d\n", *(Arr+i));
   }
}
```

Q N. 5

Program

```
#include <string.h>
#include <stdio.h>
void main()
{
   int len, i, j;
   char text[100];
   Printf("Enter string \n");
   gets(text);
   len = strlen(text);
   printf("pattern is \n");
   for (i=0; i<len; i++)
   {
     for (j=0; j<len; j++)
     {
      if ((i>=j) && (i+j>=len-1))
      {
        printf("%c ", toupper(text[j]));
      }
     } else
       {
        printf(" ");
       }
     Printf("\n");
     i = i+1;
   }
}
```

```
{
  if(i%2==0){
  printf("%c ", text[j]);
  } else {
```

Q No 4

⇒ Nested structure

Structure with in structure is called as nested structure. Nesting of structure is permitted in C. A structure variable may defined as a member of another structure. In such a situation, the declaration of embedded inner structure must appear before the declaration of other structures.

Program

```
# include <stdio.h>
# include <math.h>
# include <string.h>
struct student
{
    char name[20];
    int age;
    char faculty[20];
}
std[50], temp;
void get_info(int i){
    fflush(stdin);
    printf("\n Enter the name of the student:");
    gets(std[i].name);
    printf("Enter the age of the student:");
```

```
scanf("%d", &std[i].age);
fflush(stdin);
printf("\nEnter the faculty of the student: ");
gets(std[i].faculty);
}

void sort(int n){
  int i,j;
  for(i=0; i<n-1; i++)
  {
    for(j=0; j<n-i-1; j++)
    {
      if(strcmp(std[j].name, std[j+1].name)>0)
      {
        temp = std[j];
        std[j] = std[j+1];
        std[j+1] = temp;
      }
    }
  }
}

void show_info(int i){
  printf("\n");
  printf("The name of student is");
  puts(std[i].name);
  printf("The age of the student is ");
  printf("%d\n", std[i].age);
  printf("The faculty of the student is ");
  gets(std[i].faculty);
```

```
}
int main(){
 int num, i, count=0,n;
 char ans[10];
 while(1)
  {
   printf("Do you want to add data of the student ? In 'yes' enter any key & If not enter '\0'");
  gets(ans);
  if(strcmp(ans,"\0")==0) break;
  else{
    get_info(i);
    count++;
   }
  }
sort(count);
for(i=0; i<num; i++)
{
 show_info(i);
}
printf("Enter the nth term:");
Scanf("%d",&n);
Printf("Name: %s \t faculty: %s \t Age: %d",
std[n-1].name, std[n-1].faculty, std[n-1].age);
}
```