S.No: 25	Exp. Name: ***Illustrate the use of register variables***	Date:2023-06-26

Aim:

Illustrate the use of register variables.

- You can use the **register** storage class when you want to store local variables within functions or blocks in CPU registers instead of RAM to have quick access to these variables. For example, "counters" are a good candidate to be stored in the register.
- The keyword **register** is used to declare a register storage class. The variables declared using register storage class has lifespan throughout the program.
- It is similar to the auto storage class. The variable is limited to the particular block. The only difference is that the variables declared using register storage class are stored inside CPU registers instead of a memory. Register has faster access than that of the main memory.
- The variables declared using register storage class has no default value. These variables are often declared at the beginning of a program.
- Accessing the address of the register variables results in an error.

Try it out

```
A statement like
int *ptr = &weight;
will result in an error like
int *ptr = &weight;
address of register variable 'weight' requested
```

Follow the instructions given in the comment lines to understand the working of register variables.

Source Code:

register.c

```
#include <stdio.h>
void main() {
register int weight; // Declare a register variable weight of type int.
   printf("The default weight value is: %d\n",weight);
   weight = 65;
   printf("The current weight value is: %d\n",weight);
   // Add the line described above to obtain the error.
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
`The default weight value is: 1024482696`

S.No: 23	Exp. Name: ***Illustrate the use of auto variable***	**Date:2023-06-26**

Aim:

Illustrate the use of auto variable.
The variables defined using **auto** storage class are called as local variables.
Auto stands for **automatic** storage class. A variable is in auto storage class by default if it is not explicitly specified.
The scope of an auto variable is **limited with the particular block only.**
Once the control goes out of the block, the access is destroyed. This means only the block in which the auto variable is declared can access it.
A keyword **auto** is used to define an auto storage class. By default, an auto variable contains a **garbage value**.
Follow the instructions given in the comment lines to declare auto variables and print their values at different places in the program.

Source Code:

auto.c

```
#include<stdio.h>
void main() {
   auto int d=10;
   {
      auto int d=4;
      {
         auto int d=6;
         printf("d=%d\n",d);
      }
      printf("d=%d\n",d);
   }
   printf("d=%d\n",d);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
32767
6
4

S.No: 6	Exp. Name: ***Implement the C program which computes the sum of the first n terms of the series***	**Date:2023-06-26**

Aim:

Implement the C program which computes the sum of the first n terms of the series

Sum = 1 - 3 + 5 - 7 + 9 + ...

Sample Input and Output - 1:

```
Enter the value of n: 99
The sum of first 99 terms of the series is: 99
```

Source Code:

sum.c

```
#include<stdio.h>
void main()
{
   int n, i, sum = 0, sumn = 0, sump = 0;
   printf("Enter the value of n: ");
   scanf("%d", &n);
   for(i=0; i<n; i++)
   {
      if(i%2==0)
      {
         sump += 2*i+1;
      }
      else
      {
         sumn += -(2*i+1);
      }
   }
   sum = sump + sumn;
   printf("The sum of first %d terms of the series is: %d\n",n,sum);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
`Enter the value of n:  789`
`The sum of first 789 terms of the series is: 789`

Test Case - 2
User Output
`Enter the value of n:  76`
`The sum of first 76 terms of the series is: -76`

Test Case - 3

User Output

```
Enter the value of n:  99
The sum of first 99 terms of the series is: 99
```

S.No: 4	Exp. Name: ***Design an algorithm and implement using C language the following exchanges***	**Date:2023-06-26**

Aim:

Design an algorithm and implement using C language the following exchanges **a** ← **b** ← **c** ← **d** ← **a** and print the result as shown in the example.

```
Sample Input and Output:
Enter values of a, b, c and d: 98 74 21 36
After swapping
a = 74
b = 21
c = 36
d = 98
```

Source Code:

exchange.c

```
#include<stdio.h>
void main()
{
   int a,b,c,d,temp;
   printf("Enter values of a, b, c and d: ");
   scanf("%d %d %d %d",&a,&b,&c,&d);
   temp = a;
   a = b;
   b = c;
   c = d;
   d = temp;
   printf("After swapping\na = %d\nb = %d\nc = %d\nd = %d\n",a,b,c,d);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter values of a, b, c and d:  1 2 3 4
After swapping
a = 2
b = 3
c = 4
d = 1
```

Test Case - 2

User Output

```
Enter values of a, b, c and d:  98 74 21 36
After swapping
a = 74
b = 21
```

```
c  =  36
d  =  98
```

S.No: 2	Exp. Name: ***Write a C program to find second largest for the given numbers***	Date:2023-06-26

Aim:

Design a C program which finds the `second maximum number` among the given one dimensional array of elements.

```
Sample Input and Output:Enter how many values you want to read : 6
Enter the value of a[0] : 45
Enter the value of a[1] : 24
Enter the value of a[2] : 23
Enter the value of a[3] : 65
Enter the value of a[4] : 78
Enter the value of a[5] : 42
The second largest element of the array = 65
```

Note: Do use the **printf()** function with a **newline** character (\n) at the end.

Source Code:

second_large.c

```
#include<stdio.h>
void main()
{
   int i, n, a[20], max1=0, max2=0;
   printf("Enter how many values you want to read : ");
   scanf("%d",&n);
   for(i=0; i<n; i++)
{
   printf("Enter the value of a[%d] : ",i);
   scanf("%d", &a[i]);
}
for(i=0; i<n; i++)
{
   if(max1 < a[i])
   {
      max2 = max1;
      max1 = a[i];
   }
   else if(a[i] > max2 && a[i] < max1)
   {
      max2 = a[i];
   }
}
printf("The second largest element of the array = %d\n",max2);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
`Enter how many values you want to read :  4`
`Enter the value of a[0] :  32`

```
Enter the value of a[1] :  25
Enter the value of a[2] :  69
Enter the value of a[3] :  47
The second largest element of the array = 47
```

S.No: 1	Exp. Name: ***Write a C program to find the reverse of a given number***	**Date:2023-06-26**

Aim:

Design a C program which reverses the given number.

Source Code:

reverse.c

```
#include<stdio.h>
void main()
{
   int n,rem=0,rev=0;
   scanf("%d",&n);
   while(n>0)
   {
      rem=n%10;
      rev=rev*10+rem;
      n=n/10;
   }

printf("Reversed number= %d",rev);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
456
Reversed number= 654

Test Case - 2
User Output
958745
Reversed number= 547859