DS TUTORIAL – 2

Name: Sakshi Jain

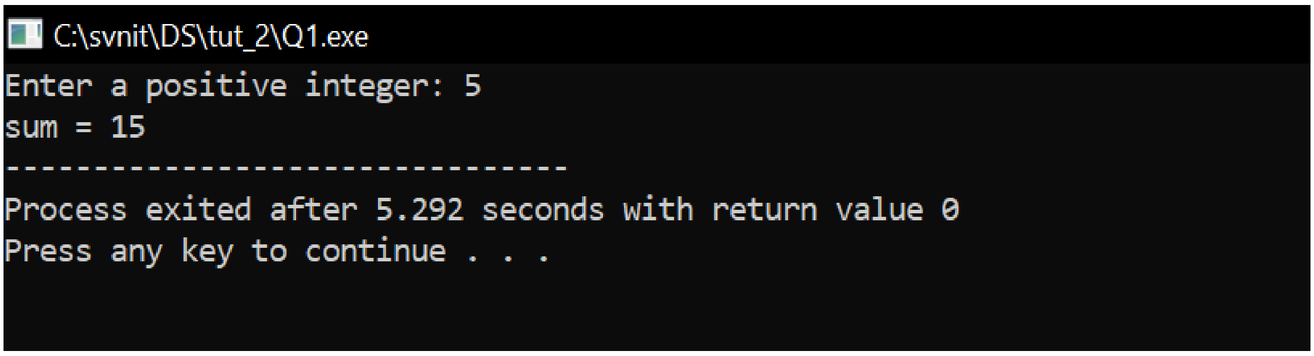
Adm No: U19CS048

1. Original code

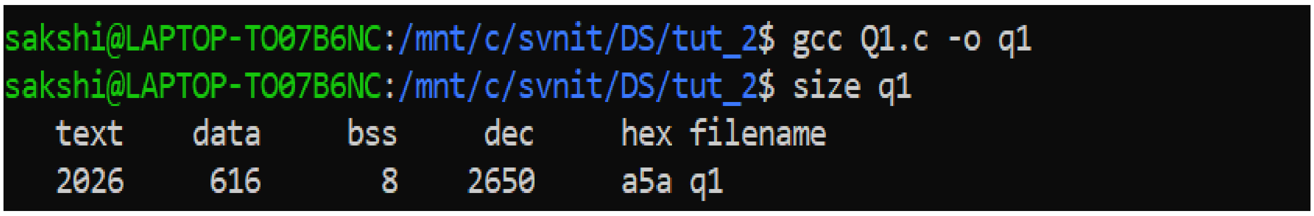
   Source code:

```
#include <stdio.h>
int sum(int n);
int main() {
   int number, result;
   printf("Enter a positive integer: ");
   scanf("%d", &number);
   result = sum(number);
   printf("sum = %d", result);
   return 0;
}
int sum(int n) {
int p=n;
   if (n != 0)
      // sum() function calls itself
      return n + sum(n-1);
   else
      return n;
}
```

   Output:



```
C:\svnit\DS\tut_2\Q1.exe
Enter a positive integer: 5
sum = 15
--------------------------------
Process exited after 5.292 seconds with return value 0
Press any key to continue . . .
```



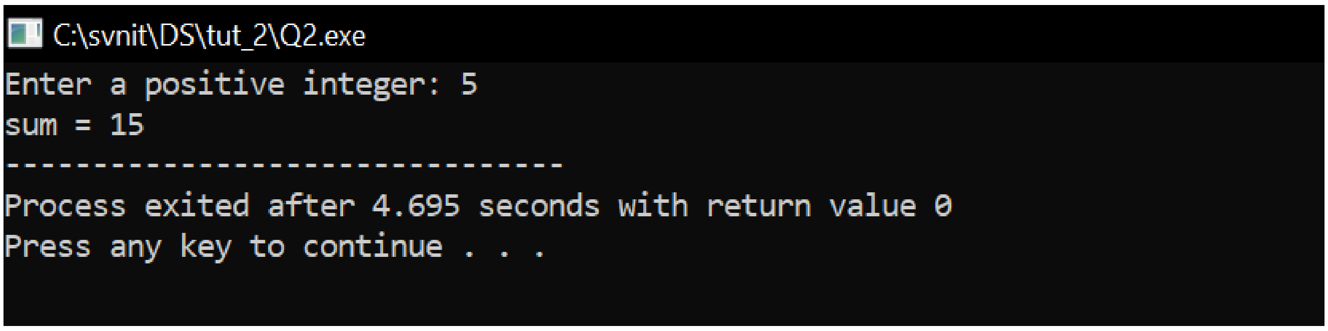
```
sakshi@LAPTOP-TO07B6NC:/mnt/c/svnit/DS/tut_2$ gcc Q1.c -o q1
sakshi@LAPTOP-TO07B6NC:/mnt/c/svnit/DS/tut_2$ size q1
   text    data     bss     dec     hex filename
   2026     616       8    2650     a5a q1
```

2. When variable “number” is declared as a Global variable

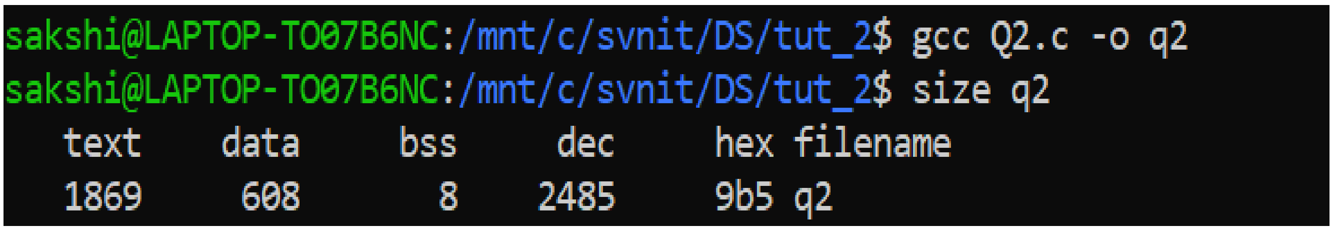
Source code:

```
#include <stdio.h>
int number;
int sum(int n);
int main() {
   int result;
   printf("Enter a positive integer: ");
   scanf("%d", &number);
   result = sum(number);
   printf("sum = %d", result);
   return 0;
}
int sum(int n) {
int p=n;
   if (n != 0)
      // sum() function calls itself
      return n + sum(n-1);
   else
      return n;
}
```

Output:



```
C:\svnit\DS\tut_2\Q2.exe
Enter a positive integer: 5
sum = 15
--------------------------------
Process exited after 4.695 seconds with return value 0
Press any key to continue . . .
```



```
sakshi@LAPTOP-TO07B6NC:/mnt/c/svnit/DS/tut_2$ gcc Q2.c -o q2
sakshi@LAPTOP-TO07B6NC:/mnt/c/svnit/DS/tut_2$ size q2
   text    data     bss     dec     hex filename
   1869     608       8    2485     9b5 q2
```

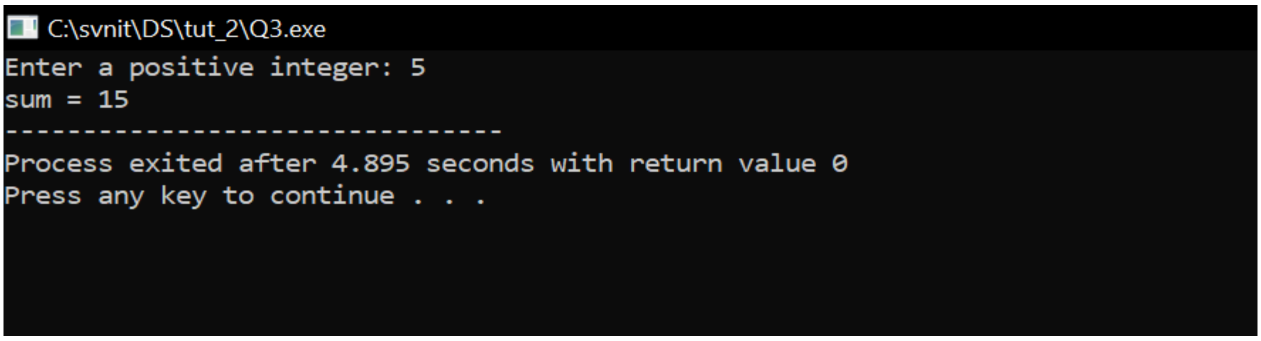
Reason – Here we declared number as a global variable but we did not initialise it so it gets stored in the uninitialized data segment (bss segment). If we initialised it, it would have been stored in data segement, so data segment would have been 612 and bss would be 4.

3. When variable “number” is declared as a Static variable

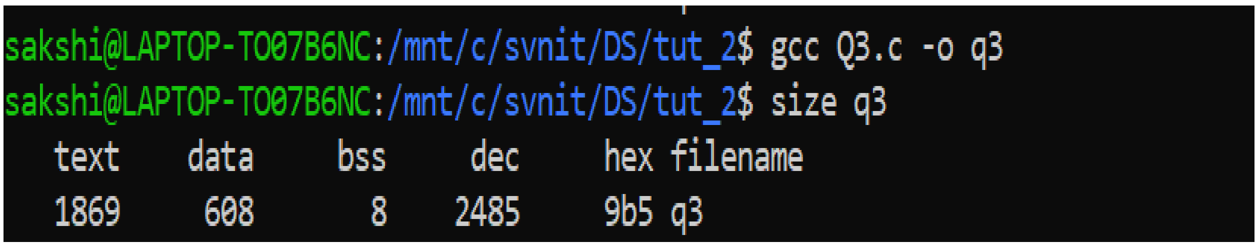
Source code:

```
#include <stdio.h>
int sum(int n);
int main() {
   static int number;
         int result;
   printf("Enter a positive integer: ");
   scanf("%d", &number);
   result = sum(number);
   printf("sum = %d", result);
   return 0;
}
int sum(int n) {
int p=n;
   if (n != 0)
      // sum() function calls itself
      return n + sum(n-1);
   else
      return n;
}
```

Output:



```
C:\svnit\DS\tut_2\Q3.exe
Enter a positive integer: 5
sum = 15
--------------------------------
Process exited after 4.895 seconds with return value 0
Press any key to continue . . .
```



```
sakshi@LAPTOP-TO07B6NC:/mnt/c/svnit/DS/tut_2$ gcc Q3.c -o q3
sakshi@LAPTOP-TO07B6NC:/mnt/c/svnit/DS/tut_2$ size q3
   text    data     bss     dec     hex filename
   1869     608       8    2485     9b5 q3
```

Reason – Here we declared number as a static variable but we did not initialise it so it gets stored in the uninitialized data segment (bss segment). If we initialised it, it would have been stored in data segement, so data segment would have been 612 and bss would be 4.

4. When variable “number” is declared as an Extern variable (Define constant in sperate header file)

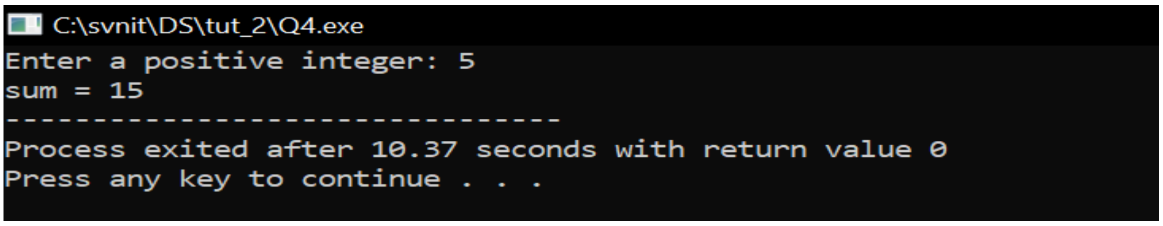
Header file:

```
int number;
```

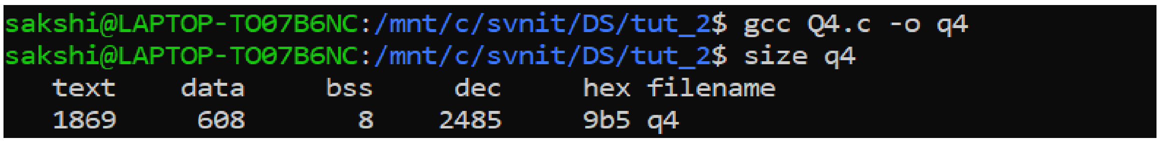
Source code:

```
#include <stdio.h>
#include "myhead.h"
int sum(int n);
extern int number;
int main() {
   int result;
   printf("Enter a positive integer: ");
   scanf("%d", &number);
   result = sum(number);
   printf("sum = %d", result);
   return 0;
}
int sum(int n) {
int p=n;
   if (n != 0)
      // sum() function calls itself
      return n + sum(n-1);
   else
      return n;
}
```

Output:



```
C:\svnit\DS\tut_2\Q4.exe
Enter a positive integer: 5
sum = 15
--------------------------------
Process exited after 10.37 seconds with return value 0
Press any key to continue . . .
```



```
sakshi@LAPTOP-TO07B6NC:/mnt/c/svnit/DS/tut_2$ gcc Q4.c -o q4
sakshi@LAPTOP-TO07B6NC:/mnt/c/svnit/DS/tut_2$ size q4
   text    data     bss     dec     hex filename
   1869     608       8    2485     9b5 q4
```

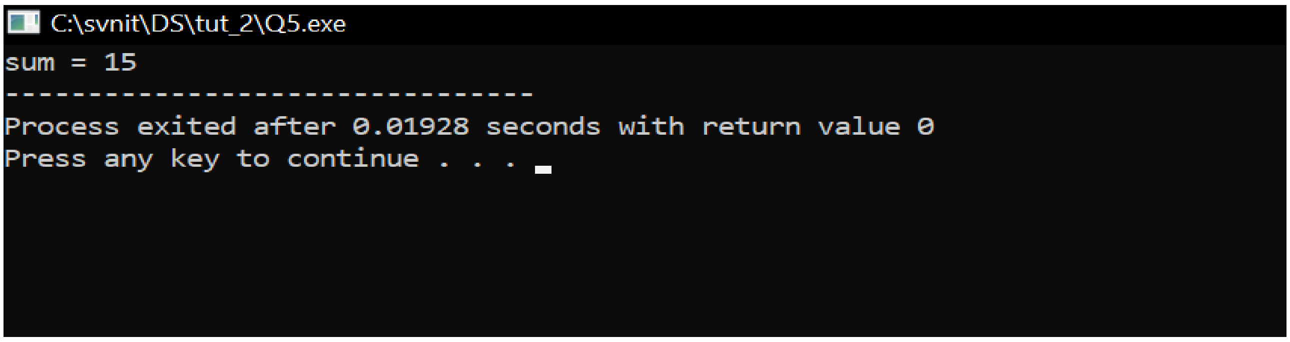
Reason - The extern keyword is used with a variable to inform the compiler that this variable is declared somewhere else. The extern declaration does not allocate storage for variables. So, an extern variable is nothing but a global variable initialized with a legal value where it is declared in order to be used elsewhere. Like global, since it is uninitialized it is stored in bss else it would have been stored in data.

5. When variable “number” is declared as a Constant variable

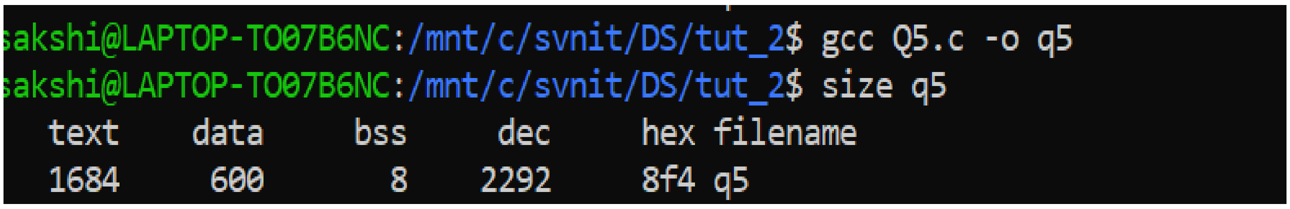
   Source code:

```
#include <stdio.h>
int sum(int n);
int main() {
   const int number=5;
   int result;
   result = sum(number);
   printf("sum = %d", result);
   return 0;
}
int sum(int n) {
int p=n;
   if (n != 0)
      // sum() function calls itself
      return n + sum(n-1);
   else
      return n;
}
```

   Output:



```
C:\svnit\DS\tut_2\Q5.exe
sum = 15
--------------------------------
Process exited after 0.01928 seconds with return value 0
Press any key to continue . . .
```



```
sakshi@LAPTOP-TO07B6NC:/mnt/c/svnit/DS/tut_2$ gcc Q5.c -o q5
sakshi@LAPTOP-TO07B6NC:/mnt/c/svnit/DS/tut_2$ size q5
   text    data     bss     dec     hex filename
   1684     600       8    2292     8f4 q5
```

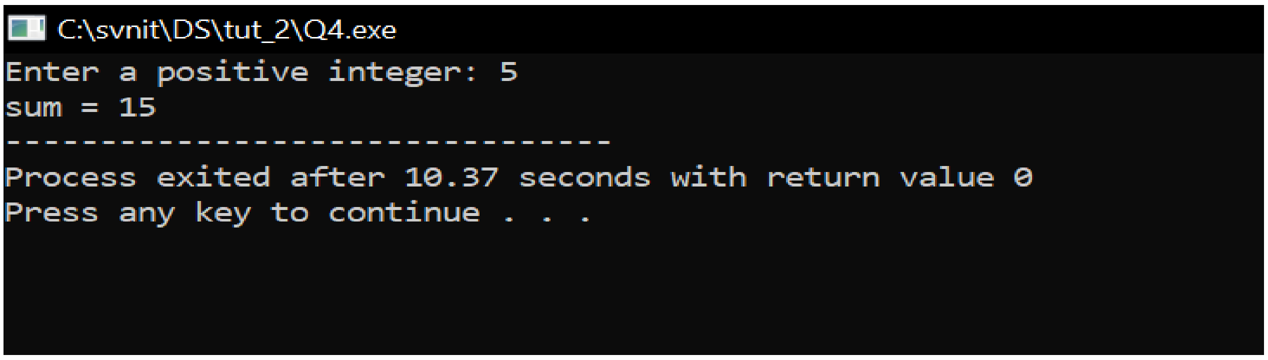
Reason – Here constant variable is initialised and so it is stored in the read-write area of the stack. When it is uninitialized it is stored in the text segment.

6. When variable “number” is declared as an Auto variable

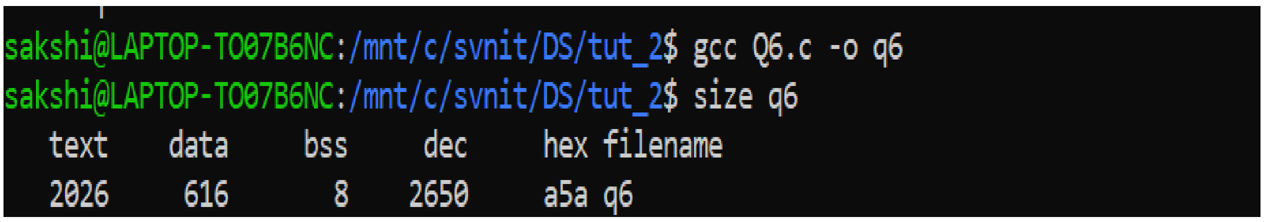
Source code:

```
#include <stdio.h>
int sum(int n);
int main() {
   auto int number;
         int result;
   printf("Enter a positive integer: ");
   scanf("%d", &number);
   result = sum(number);
   printf("sum = %d", result);
   return 0;
}
int sum(int n) {
int p=n;
   if (n != 0)
      // sum() function calls itself
      return n + sum(n-1);
   else
      return n;
}
```

Output:



```
C:\svnit\DS\tut_2\Q4.exe
Enter a positive integer: 5
sum = 15
--------------------------------
Process exited after 10.37 seconds with return value 0
Press any key to continue . . .
```



```
sakshi@LAPTOP-T007B6NC:/mnt/c/svnit/DS/tut_2$ gcc Q6.c -o q6
sakshi@LAPTOP-T007B6NC:/mnt/c/svnit/DS/tut_2$ size q6
   text    data     bss     dec     hex filename
   2026     616       8    2650     a5a q6
```

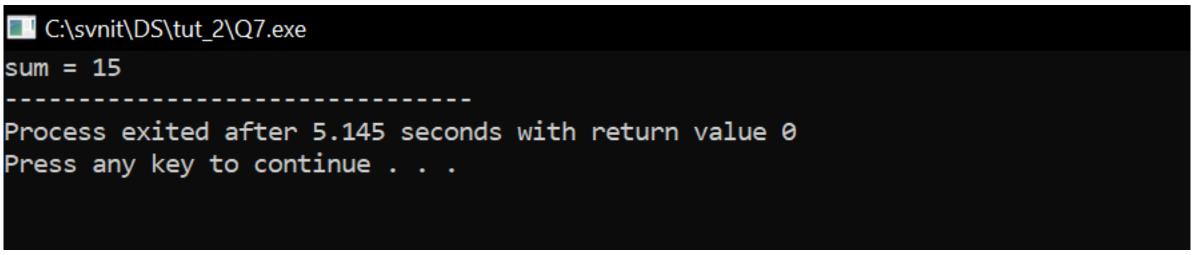
Reason – Here we declared number as auto variable, they are always local and stored on the stack independent of whether they are initialised or not. So data and bss value will always be same whether number is initialised or it is not.

7. When variable “number” is declared as a Register variable

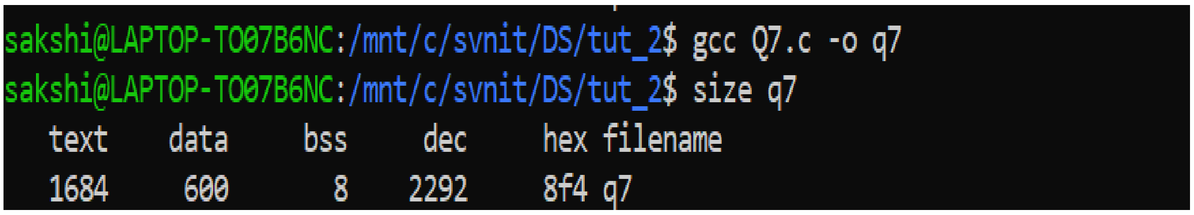
Source code:

```
#include <stdio.h>
int sum(int n);
int main() {
   register int number=5;
   int result;
   result = sum(number);
   printf("sum = %d", result);
   return 0;
}
int sum(int n) {
int p=n;
   if (n != 0)
      // sum() function calls itself
      return n + sum(n-1);
   else
      return n;
}
```

Output:



```
C:\svnit\DS\tut_2\Q7.exe
sum = 15
--------------------------------
Process exited after 5.145 seconds with return value 0
Press any key to continue . . .
```



```
sakshi@LAPTOP-T007B6NC:/mnt/c/svnit/DS/tut_2$ gcc Q7.c -o q7
sakshi@LAPTOP-T007B6NC:/mnt/c/svnit/DS/tut_2$ size q7
   text    data     bss     dec     hex filename
   1684     600       8    2292     8f4 q7
```

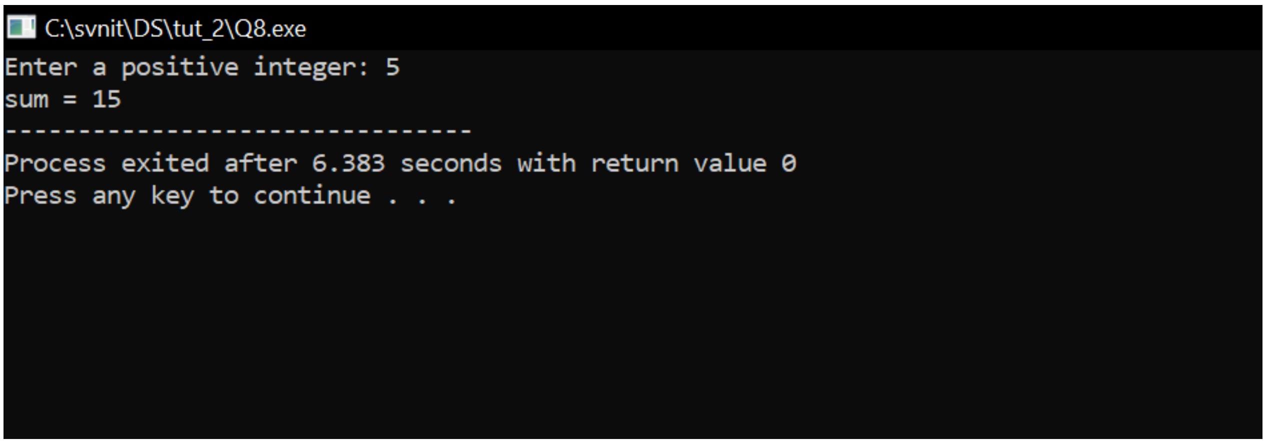
Reason – When variable is declared as register, compiler tries to store these variables in the register of the microprocessor if a free register is available. Register variables have faster accessibility than a normal variable. Generally, the frequently used variables are kept in registers. If a free register is not available, these are then stored in the memory only, in such a scenario it works like an auto variable. So comparing to global and static, the data and the bss segments of register variables occupy less space.

8. When variable “p” is declared as an Auto variable

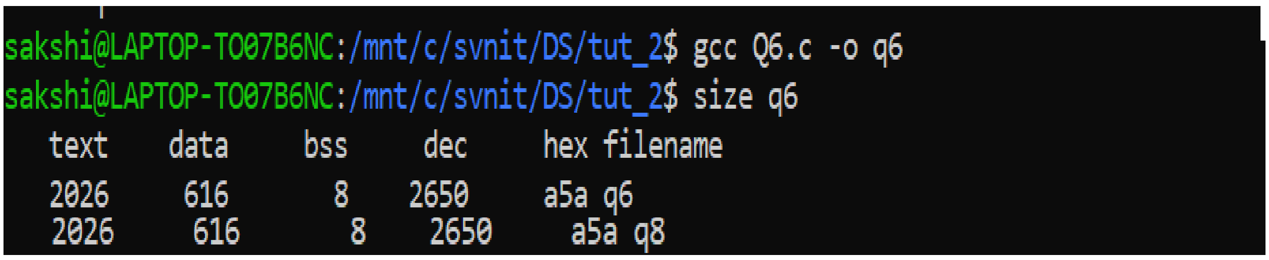
Source code:

```
#include <stdio.h>
int sum(int n);
int main() {
   int number, result;
   printf("Enter a positive integer: ");
   scanf("%d", &number);
   result = sum(number);
   printf("sum = %d", result);
   return 0;
}
int sum(int n) {
auto int p=n;
   if (n != 0)
      // sum() function calls itself
      return n + sum(n-1);
   else
      return n;
}
```

Output:



```
C:\svnit\DS\tut_2\Q8.exe
Enter a positive integer: 5
sum = 15
--------------------------------
Process exited after 6.383 seconds with return value 0
Press any key to continue . . .
```



```
sakshi@LAPTOP-T007B6NC:/mnt/c/svnit/DS/tut_2$ gcc Q6.c -o q6
sakshi@LAPTOP-T007B6NC:/mnt/c/svnit/DS/tut_2$ size q6
   text    data     bss     dec     hex filename
   2026     616       8    2650     a5a q6
   2026     616       8    2650     a5a q8
```

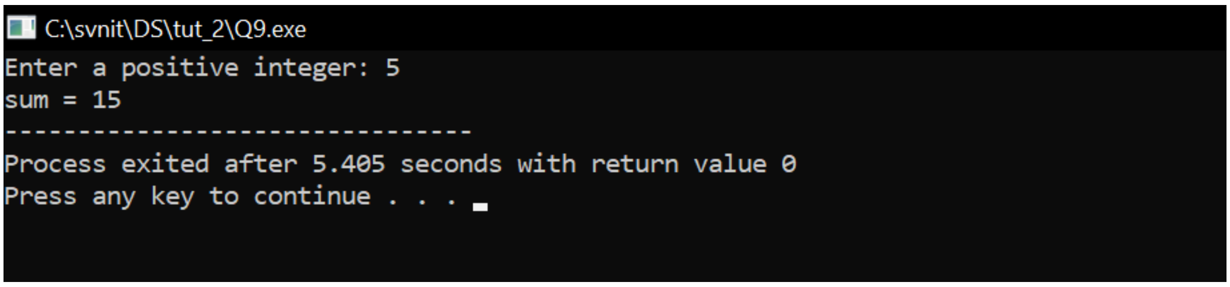
Reason - Here we declared p as auto variable, they are always local and stored on the stack independent of whether they are initialised or not. So data and bss value will always be same whether number is initialised or it is not.

9. When variable “p” is declared as a Static variable

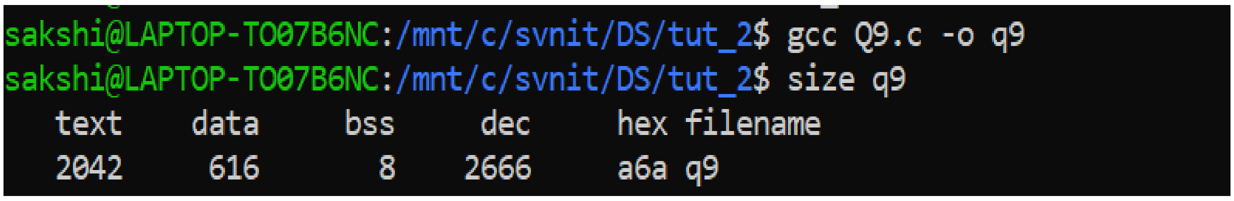
Source code:

```
#include <stdio.h>
int sum(int n);
int main() {
   int number, result;
   printf("Enter a positive integer: ");
   scanf("%d", &number);
   result = sum(number);
   printf("sum = %d", result);
   return 0;
}
int sum(int n) {
static int p;
   if (n != 0)
      // sum() function calls itself
      return n + sum(n-1);
   else
      return n;
}
```

Output:



```
C:\svnit\DS\tut_2\Q9.exe
Enter a positive integer: 5
sum = 15
--------------------------------
Process exited after 5.405 seconds with return value 0
Press any key to continue . . .
```



```
sakshi@LAPTOP-TO07B6NC:/mnt/c/svnit/DS/tut_2$ gcc Q9.c -o q9
sakshi@LAPTOP-TO07B6NC:/mnt/c/svnit/DS/tut_2$ size q9
   text    data     bss     dec     hex filename
   2042     616       8    2666     a6a q9
```

Reason – Here we declared p as a static variable but we did not initialise it so it gets stored in the uninitialized data segment (bss segment). If we initialised it, it would have been stored in data segement, so data segment would have been 620 and bss would be 4. Here, the local scope of static variable doesn’t affect the way it is stored.