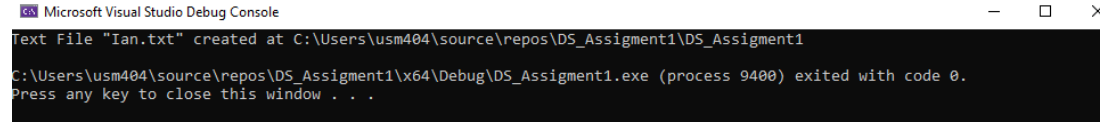
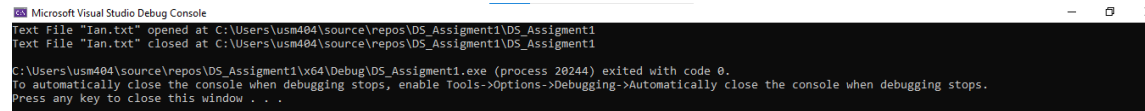


Question 1 Screenshot



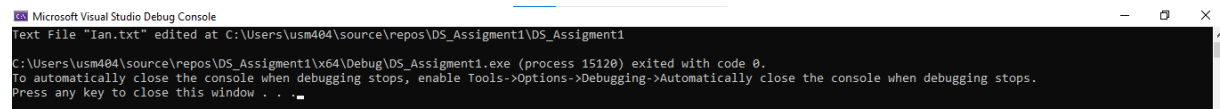
```
Microsoft Visual Studio Debug Console
Text File "Ian.txt" created at C:\Users\usm404\source\repos\DS_Assignment1\DS_Assignment1
C:\Users\usm404\source\repos\DS_Assignment1\x64\Debug\DS_Assignment1.exe (process 9400) exited with code 0.
Press any key to close this window . . .
```

Question 2 Screenshot



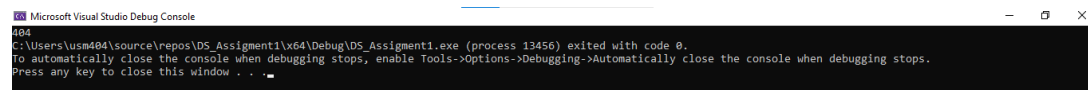
```
Microsoft Visual Studio Debug Console
Text File "Ian.txt" opened at C:\Users\usm404\source\repos\DS_Assignment1\DS_Assignment1
Text File "Ian.txt" closed at C:\Users\usm404\source\repos\DS_Assignment1\DS_Assignment1
C:\Users\usm404\source\repos\DS_Assignment1\x64\Debug\DS_Assignment1.exe (process 20244) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

Question 3 Screenshot



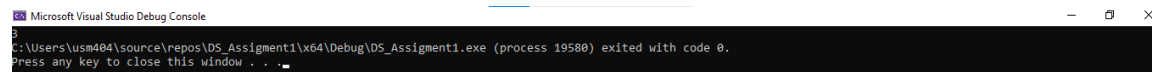
```
Microsoft Visual Studio Debug Console
Text File "Ian.txt" edited at C:\Users\usm404\source\repos\DS_Assignment1\DS_Assignment1
C:\Users\usm404\source\repos\DS_Assignment1\x64\Debug\DS_Assignment1.exe (process 15120) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

Question 4 Screenshot



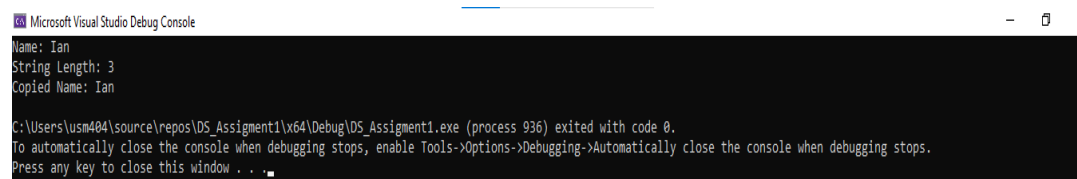
```
Microsoft Visual Studio Debug Console
404
C:\Users\usm404\source\repos\DS_Assignment1\x64\Debug\DS_Assignment1.exe (process 13456) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

Question 5 Screenshot



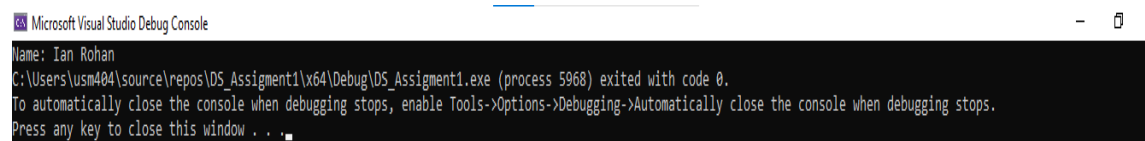
```
Microsoft Visual Studio Debug Console
3
C:\Users\usm404\source\repos\DS_Assignment1\x64\Debug\DS_Assignment1.exe (process 19580) exited with code 0.
Press any key to close this window . . .
```

Question 6 Screenshot



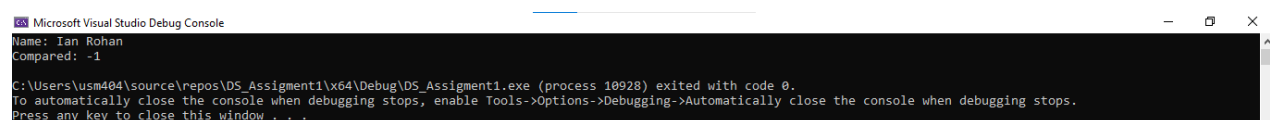
```
Microsoft Visual Studio Debug Console
Name: Ian
String Length: 3
Copied Name: Ian
C:\Users\usm404\source\repos\DS_Assignment1\x64\Debug\DS_Assignment1.exe (process 936) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

Question 7 Screenshot



```
Microsoft Visual Studio Debug Console
Name: Ian Rohan
C:\Users\usm404\source\repos\DS_Assignment1\x64\Debug\DS_Assignment1.exe (process 5968) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

Question 8 Screenshot



```
Microsoft Visual Studio Debug Console
Name: Ian Rohan
Compared: -1
C:\Users\usm404\source\repos\DS_Assignment1\x64\Debug\DS_Assignment1.exe (process 10928) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

Question 9 Screenshot

```
Microsoft Visual Studio Debug Console
Last Name: Rohan
Converted Name: rohan
C:\Users\usm404\source\repos\DS_Assignment1\x64\Debug\DS_Assignment1.exe (process 9912) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

Question 10 Screenshot

```
Microsoft Visual Studio Debug Console
Last Name: Rohan
Converted Name: ROHAN
C:\Users\usm404\source\repos\DS_Assignment1\x64\Debug\DS_Assignment1.exe (process 13336) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

Question 11 Screenshot

```
Enter number of elements(Malloc): 3
Enter Elements:
4
0
4

Memory Address of 4 is 0x5628810beac0
Memory Address of 0 is 0x5628810beac4
Memory Address of 4 is 0x5628810beac8

...Program finished with exit code 0
Press ENTER to exit console.
```

Question 12 Screenshot

```
Enter number of elements(Calloc): 3
Enter Elements:
4
0
4

Memory Address of 4 is 0x5587d6203ac0
Memory Address of 0 is 0x5587d6203ac4
Memory Address of 4 is 0x5587d6203ac8

...Program finished with exit code 0
Press ENTER to exit console.
```

Question 13: Describe the difference between the two types of memory allocation i.e. malloc and calloc.

The differences between malloc and calloc memory allocation are the arguments they can take and how they initialize memory content. Malloc takes a single argument representing the total number of bytes to allocate and does not initialize the memory content. In comparison, calloc takes two arguments: the number of blocks and the size of each block in bytes and initializes the allocated memory to zero. Using one or the other depends on your specific needs. For example, if your project requires zero-initialized memory, calloc would be the better choice, otherwise malloc could be more convenient.

Question 14: Explain pointers and pointers to pointers using code, create code to explain both as examples

Pointers are variables that store the memory addresses of other variables. Being able to store such data allows programmers to manipulate the data in the computer's memory, allowing for more efficient programs and the creation of dynamic data structures.

```
1  #include <stdio.h>
2
3  int main() {
4      // Pointer example
5      // Ian Rohan
6
7      // Declare a variable and initialize it
8      int x = 10;
9
10     // Declare a pointer and initialize it with the address of the variable
11     int* ptr = &x;
12
13     // Access the value using the pointer
14     printf("Value of x: %d\n", *ptr);
15
16     // Modify the value using the pointer
17     *ptr = 20;
18
19     // Print the modified value
20     printf("Modified value of x: %d\n", x);
21
22     // Print the memory address stored in the pointer
23     printf("Memory address stored in the pointer: %p\n", (void*)ptr);
24
25     return 0;
26 }
27
```

```
Value of x: 10
Modified value of x: 20
Memory address stored in the pointer: 00000094318FFBB4
```

In this example, a pointer has been initialized with the memory address of the variable “x”. From this, the pointer variable can access the stored value at said memory address, being able to call it or modify it later.

Pointers to pointers store the memory address of another pointer. This is typically seen when needing to dynamically allocate memory for a pointer or when dealing with multi-dimensional arrays.

```
1  #include <stdio.h>
2
3  int main() {
4      // Pointer to pointer example
5      // Ian Rohan
6
7      int x = 10;
8      int* ptr = &x;
9      int** pptr = &ptr;
10
11     printf("Value of x: %d\n", **pptr);
12
13     // Modifying the value using the pointer to pointer
14     **pptr = 20;
15     printf("Modified value of x: %d\n", x);
16
17     return 0;
18 }
```

```
Value of x: 10
Modified value of x: 20
```

In this example, a pointer of x is created. In addition to this, another pointer is initialized with the memory address of the previously mentioned pointer, as seen with the double asterisk (**). This is also known as a double pointer (hence the double asterisk). From this code, the double pointer accesses the stored value of the first pointer, which in turn accesses the stored value of x, allowing the double pointer to call it or modify later. As stated previously, the use of double pointers comes into fruition when dealing with dynamic memory allocation and multi-dimensional arrays.