**Answer Script**

|  |
| --- |
| Question No. 01 |
| Write a C program to take positive integer **N** as input and print a pattern shown in the sample input output. **Marks**: 20  **Constraints**: 1 <= **N** <= 5 |
| Answer No. 01 |
| #include <stdio.h>  int main()  {      int n, s, k;      scanf("%d", &n);      s = n - 1;      k = 1;      for (int i = 1; i <= (2 \* n) - 1; i++)      {          for (int j = 1; j <= s; j++)          {              printf(" ");          }          for (int j = 1; j <= k; j++)          {              printf("%d", j);          }          if (i <= n - 1)          {              s--;              k = k + 2;          }          else          {              s++;              k = k - 2;          }          printf("\n");      }      return 0;  } |

|  |
| --- |
| Question No. 02 |
| Write a C program to take positive integer **N** as input and print a pattern shown in the sample input output. |
| Answer No. 02 |
| #include <stdio.h>  int main()  {      int n, s, k;      scanf("%d", &n);      s = n - 1;      k = 1;      for (int i = 1; i <= n; i++)      {          for (int j = s; j > 0; j--)          {              printf(" ");          }          for (int j = 1; j <= k; j++)          {              printf("%d", j);          }          if (i <= n - 1)          {              s--;              k++;          }          else          {              s++;              k = k - 2;          }          printf("\n");      }      return 0;  } |

|  |
| --- |
| Question No. 03 |
| Write a function named **count\_before\_zero()** which receives an array of integers and the size of that array and counts the number of elements in that array until you find zero and returns that count. Call that function in the main function and print the count there |
| Answer No. 03 |
| #include <stdio.h>  int count\_before\_zero(int arr[], int size)  {      int count = 0;      for (int i = 0; i < size; i++)      {          if (arr[i] == 0)          {              break;          }          count++;      }      return count;  }  int main()  {      int n;      scanf("%d", &n);      int arr[n];      for (int i = 0; i < n; i++)      {          scanf("%d", &arr[i]);      }      int sz = sizeof(arr) / sizeof(int);      int count = count\_before\_zero(arr, sz);      printf("%d\n", count);      return 0;  } |

|  |
| --- |
| Question No. 04 |
| Show the 4 types of examples of functions given below with an example. You can give any example you want, but make sure you are giving different examples for all the four types |
| Answer No. 04 |
| // Has Return + Parameter:  int add(int x, int y){      return x + y;  }  // Has Return + No Parameter:  int sum()  {      int a, b;      scanf("%d %d", &a, &b);      int s = a + b;        return s;  }  //No Return + Parameter:  void sum(int x, int y)  {      int s = x + y;      printf("%d\n", s);  }  // No Return + No Parameter:  void Message(){      printf("Hello world!");  } |

|  |
| --- |
| Question No. 05 |
| Write a function named **is\_palindrome()** which will receive a string as parameter from the main function and this function will return 1 if the string is palindrome, otherwise it will return 0. And with the help of this 1 or 0 print “Palindrome” or “Not Palindrome” in the main function. |
| Answer No. 05 |
| #include <stdio.h>  #include <string.h>  int is\_palindrome(char s[])  {      int i = 0, j = strlen(s) - 1;      int pal = 1;      while (i < j)      {          if (s[i] != s[j])          {              pal = 0;              break;          }          i++;          j--;      }      return pal;  }  int main()  {      char s[1001];      scanf("%s", s);      if (is\_palindrome(s) == 1)      {          printf("Palindrome\n");      }      else      {          printf("Not Palindrome\n");      }      return 0;  } |

|  |
| --- |
| Question No. 06 |
| Explain about **pass by value** and **pass by reference** with an example. |
| Answer No. 06 |
| // Here is an example of pass by value:  /\*  In pass by value, a copy of the parameter is passed to the function. This means that any  changes made to the parameter inside the function will not affect the original value of  the parameter outside the function.  \*/  #include <stdio.h>  void Add(int x) {      x = x + 10;      printf("Inside function: %d\n", x);  }  int main() {      int x = 5;      printf("Before function: %d\n", x);      Add(x);      printf("After function: %d\n", x);      return 0;  }  /\*  we define a function named add that takes an integer parameter a. Inside the function, we add 10 to a and print its new value. In the main function, we define an integer variable a and initialize it to 5. We then print the value of a before calling the Add function.  Next, we call the add function and pass the value of a as an parameter. Inside the Add function, the value of a is copied into a temporary variable, and the function works with this copy. When the function returns, the temporary variable is destroyed, and the original value of a in the main function is not modified. Finally, we print the value of a after calling the add function, and we can see that it is still 5.  \*/  // Here's an example of pass by reference:  /\*  A reference to the parameter is passed to the function. This means that any changes made to the parameter inside the function will affect the original value of the parameter outside the function.  \*/  #include <stdio.h>  void Add(int \*ptr) {      \*ptr = \*ptr + 10;      printf("Inside function: \*ptr = %d\n", \*ptr);  }  int main() {      int x = 5;      int \*ptr = &x;      printf("Before function:  %d\n", x);      Add(ptr);      printf("After function:  %d\n", x);      return 0;  }  /\*  we declare a pointer variable ptr that points to the memory address of a. We pass ptr to the function Add() which takes a pointer as its parameter. Inside the function, we dereference the pointer using the \* operator to access the value stored at the memory address that ptr points to. We then add 10 to that value and store it back in the same memory location. When we return to the main function, the value of a has been changed to 15 because we modified it directly in memory using pass by reference.  \*/ |