**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;  scanf("%d", &n);  int space = n - 1;  int digit = 1;  for (int i = 1; i <= (2 \* n - 1); i++)  {  for (int j = 1; j <= space; j++)  {  printf(" ");  }  for (int j = 1; j <= digit; j++)  {  printf("%d", j);  }  if (i < n)  {  space--;  digit += 2;  }  else  {  space++;  digit -= 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.  **Marks**: 20  **Constraints**: 1 <= **N** <= 9 |
| Answer No. 02 |
| #include <stdio.h>  int main()  {  int n;  scanf("%d", &n);  int space = n - 1;  for (int i = 1; i <= n; i++)  {  for (int j = 1; j <= space; j++)  {  printf(" ");  }  for (int j = 1; j <= i; j++)  {  printf("%d", i);  }  space--;  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.  **Marks**: 15 |
| 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 result = count\_before\_zero(arr, n);  printf("%d", result);  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..  **Marks**: 20   1. Has Return + Parameter 2. Has Return + No Parameter 3. No Return + Parameter 4. No Return + No Parameter |
| Answer No. 04 |
| // 01. This function has both return and parameter  #include <stdio.h>  int summation(int a, int b)  {  int sum = a + b;  return sum;  }  int main()  {  int a, b;  scanf("%d %d", &a, &b);  int result = summation(a, b);  printf("Sum = %d\n", result);  return 0;  }  // 02. This function has parameter but no return  #include <stdio.h>  void even\_odd(int a)  {  if (a % 2 == 0)  {  printf("Even\n");  }  else  {  printf("Odd\n");  }  }  int main()  {  int a;  scanf("%d", &a);  even\_odd(a);  return 0;  }  // 03. This function has return but no parameter  #include <stdio.h>  #include <limits.h>  int findMax(void)  {  int count;  scanf("%d", &count);  int arr[count];  int max = INT\_MIN;  for (int i = 0; i < count; i++)  {  scanf("%d", &arr[i]);  if (arr[i] > max)  {  max = arr[i];  }  }  return max;  }  int main()  {  int result = findMax();  printf("%d", result);  return 0;  }  // 04. This function has no parameter and no return  #include <stdio.h>  #include <limits.h>  void findMin(void)  {  int count;  scanf("%d", &count);  int arr[count];  int Min = INT\_MAX;  for (int i = 0; i < count; i++)  {  scanf("%d", &arr[i]);  if (arr[i] < Min)  {  Min = arr[i];  }  }  printf("%d", Min);  }  int main()  {  findMin();  return 0;  } |

| Question No. 05 |
| --- |
| **Question:** 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.  **Marks**: 15  **Constraints**: String length will be maximum 10. |
| Answer No. 05 |
| #include <stdio.h>  #include <string.h>  int is\_palindrome(char str[])  {  int len = strlen(str);  for (int i = 0; i < len / 2; i++)  {  if (str[i] != str[len - 1 - i])  {  return 0; // not palindrome  }  }  return 1; // palindrome  }  int main()  {  char str[100];  scanf("%s", str);  if (is\_palindrome(str))  {  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 |
| In **pass by value**, the value of a function parameter is copied to another location of the memory. When accessing or modifying the variable within the function, it accesses only the copy. Thus, there is no effect on the original value.  **Example:**  #include <stdio.h>  void swap(int x, int y)  {  int t;  t = x;  x = y;  y = t;  }  int main()  {  int m = 10, n = 20;  printf("Before Swapping m = %d\t n = %d\n", m, n);  swap(m, n);  printf("After Swapping m = %d\t n = %d\n", m, n);  return 0;  }  In pass by reference, the memory address is passed to a function using a parameter of a pointer variable. The pointer variable holds memory address of another variable and which is used later for accessing or dereferencing that variable.  **Example:**  **#include <stdio.h>**  **void swap(int \*x, int \*y)**  **{**  **int t;**  **t = \*x;**  **\*x = \*y;**  **\*y = t;**  **}**  **int main()**  **{**  **int m = 10, n = 20;**  **printf("Before Swapping m = %d\t n = %d\n", m, n);**  **swap(&m, &n);**  **printf("After Swapping m = %d\t n = %d\n", m, n);**  **return 0;**  **}**  **Difference:**  **Pass by value** refers to a mechanism of copying the function parameter value to another variable while the **pass by reference** refers to a mechanism of passing the actual parameters to the function using memory address. This is the main difference between pass by value and pass by reference. |