**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. |
| Answer No. 01 |
| #include<stdio.h>  int main ()  {      int n;      scanf("%d", &n);      int s = n-1;      int k = 1;      for (int j = 1; j <= ((2\*n) - 1 ); j++)      {          for (int i = 1; i <= s; i++)          {              printf(" ");          }          for (int i = 1; i <= k; i++)          {              printf("%d", i);          }          if (j <= 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;      scanf("%d", &n);      int s=n-1;      int k=1;      for (int i = 1; i <= n; i++)      {          for (int j = 1; j <= s; j++)          {              printf(" ");          }          for (int j = 1; j <= k; j++)          {              printf("%d", i);          }          s--;          k++;          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 cnt = 0;      for (int i = 0; i < size; i++)      {          if (arr[i] == 0)          {              break;          }          else          {              cnt++;          }      }      return cnt;    }  int main ()  {      int n;      scanf("%d", &n);      int arr[n];      for (int i = 0; i < n; i++)      {          scanf("%d", &arr[i]);      }      int count = count\_before\_zero(arr, n);      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  #include<stdio.h>  int add(int a, int b){      int sum = a+b;      return sum;  }  int main ()  {      int x, y;      scanf("%d %d", &x, &y);      int result = add(x, y);      printf("%d\n", result);      return 0;  }    //Has Return + No Parameter  #include<stdio.h>  int subtraction(void){      int x, y;      scanf("%d %d", &x, &y);      int sub = x-y;      return sub;  }  int main ()  {      int result = subtraction();      printf("%d\n", result);      return 0;  }    // No Return + Parameter  #include<stdio.h>  void dot(int a, int b){        int result = a\*b;      printf("%d\n", result);      return;  }  int main ()  {      int x, y;      scanf("%d %d", &x, &y);      dot(x, y);      return 0;  }    // No Return + No Parameter  #include<stdio.h>  void reminder(void){        int x, y;      scanf("%d %d", &x, &y);      int result = x%y;      printf("%d\n", result);      return;  }  int main ()  {      reminder();      return 0;  } |

|  |
| --- |
| 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 \*seq){      char orginal\_seq[10];      char reverse\_seq[10];      int len = strlen(seq);      int result = 1;        for (int i = 0; i <= len; i++)      {          if (i==len)          {              reverse\_seq[i] = '\0';              orginal\_seq[i] = '\0';          } else {              orginal\_seq[i] = seq[i];              reverse\_seq[i] = seq[len-i-1];          }        }      for (int i = 0; i < len; i++)      {          if (orginal\_seq[i] != reverse\_seq[i])          {              result = 0;              break;          }        }      return result;  }  int main ()  {      char seq[10];      scanf("%s", &seq);        int result = is\_palindrome(seq);      if (result == 1)      {          printf("Palindrome");      } else {          printf("Not Palindrome");      }      return 0;    } |

|  |
| --- |
| Question No. 06 |
| Explain about **pass by value** and **pass by reference** with an example. |
| Answer No. 06 |
| **Pass By Value –** In Pass By Value, value of a variable is passed to a function as parameter. This original value will be copied to another variable of that function. The memory address of two variable will be different. If we change the value of the variable from the function, it will not change the original value or the first said variable. That means, in this case we can not access the original variable.  #include<stdio.h>  void pass\_by\_value(int n){      n = n + 10;      printf("Value of n in pass\_by\_value function - %d\n", n);  }  int main ()  {      int n;      scanf("%d", &n);      printf("Value of n in main function, before passing - %d\n", n);      pass\_by\_value(n);      printf("Value of n in main function, after passing - %d\n", n);      return 0;  }  From the output of the code, it has been see that, value of n doesn’t change although the value of that is modified in pass\_by\_value function.  **Pass By Reference** – In pass by reference, memory address of the variable is passed to the function as parameter. If we change the value of the variable from the function, it will also change the original value or the first said variable. That means, in this case we can access the original variable.  #include<stdio.h>  void pass\_by\_reference(int \*n){      \*n = \*n + 10;      printf("Value of n in pass\_by\_reference function - %d\n", \*n);  }  int main ()  {      int n;      scanf("%d", &n);      int \*ptr = &n;      printf("Value of n in main function, before passing - %d\n", n);      pass\_by\_reference(ptr);      printf("Value of n in main function, after passing - %d\n", n);      return 0;  }  From the output of the code, it has been see that, value of n is changing when the value of that is modified in pass\_by\_reference function. |