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

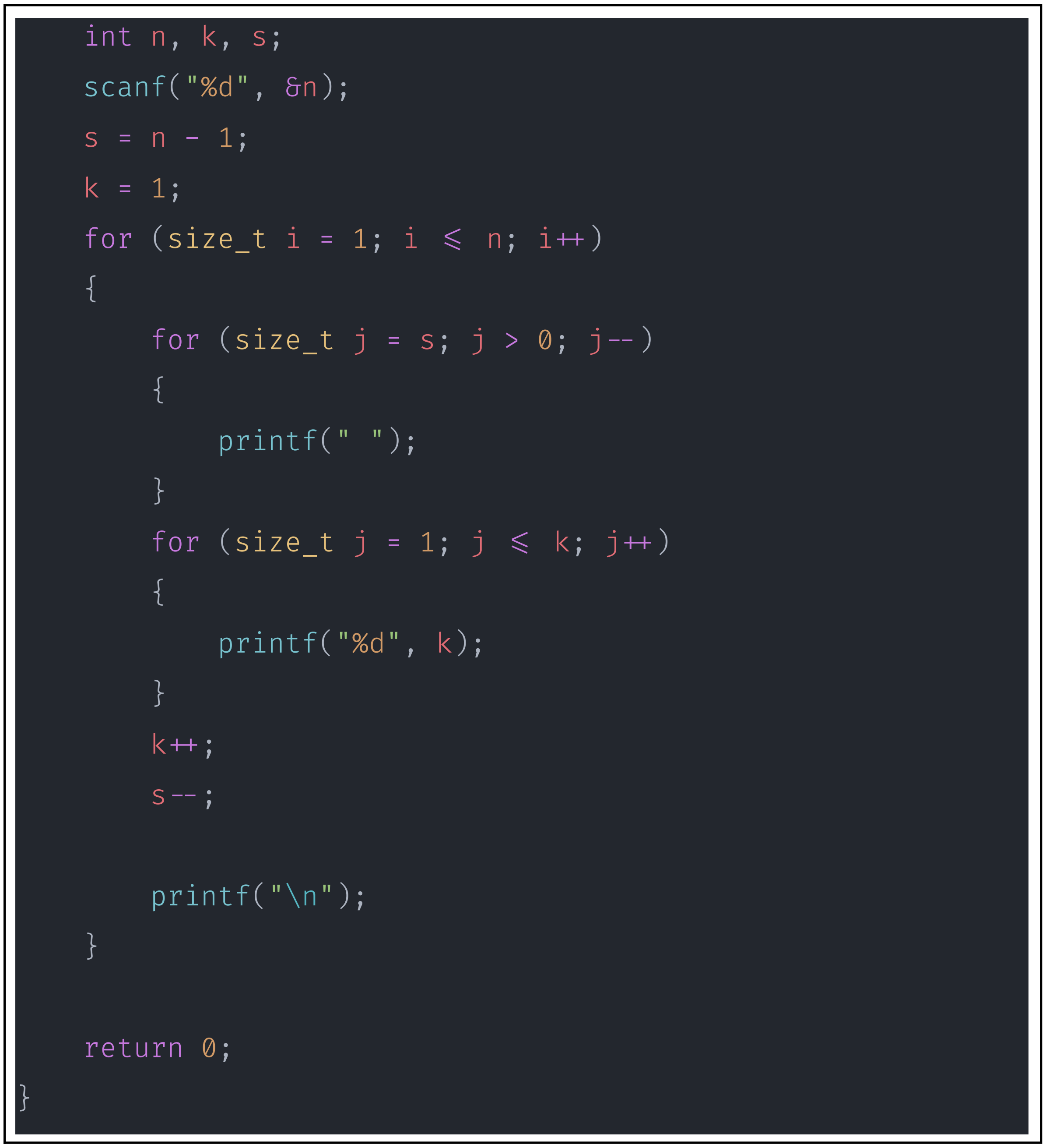
```c
#include <stdio.h>

int main()
{
    int n, k, s;
    scanf("%d", &n);
    s = n - 1;
    k = 1;

    for (size_t i = 1; i <= (2 * n) - 1; i++)
    {
        for (size_t j = 1; j <= s; j++)
        {
            printf(" ");
        }
        for (size_t j = 1; j <= k; j++)
        {
            printf("%d", j);
        }
        if (i <= n - 1)
        {
            k += 2;
            s--;
        }
        else
        {
            k -= 2;
            s++;
        }
        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()
{
```



```c
    int n, k, s;
    scanf("%d", &n);
    s = n - 1;
    k = 1;
    for (size_t i = 1; i <= n; i++)
    {
        for (size_t j = s; j > 0; j--)
        {
            printf(" ");
        }
        for (size_t j = 1; j <= k; j++)
        {
            printf("%d", k);
        }
        k++;
        s--;

        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

```c
#include <stdio.h>
int count_before_zero(int *arr, int sz)
{
    int count = 0;
    for (int i = 0; i < sz; i++)
    {
        if (arr[i] != 0)
        {
            count++;
        }
        else
        {
            return count;
        }
    }
}

int main()
{
    int sz;
    scanf("%d", &sz);
    int arr[sz];
    for (size_t i = 0; i < sz; i++)
    {
        scanf("%d", &arr[i]);
    }
    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.

1. Has Return + Parameter
2. Has Return + No Parameter
3. No Return + Parameter
4. No Return + No Parameter

Answer No. 04

1.Has Return + Parameter:

```c
#include <stdio.h>
int add(int a, int b)
{
    return a + b;
}


int main()
{
    int sum = add(3, 5); // sum == 8

    return 0;
}
```

2.Has Return + No Parameter:

```
#include <stdio.h>

int get_random_number()

{

    return rand();

}



int main()

{

    int num = get_random_number(); // num will be a random
integer



    return 0;
}
```

3.No Return + Parameter:

```
#include <stdio.h>

void print_hello(char *name)

{

    printf("Hello, %s!\n", name);

}



int main()

{

    print_hello("Phitron"); // prints "Hello, phitron!"

    return 0;

}
```

4.No Return + No Parameter:

```
#include <stdio.h>

void greet()

{
```

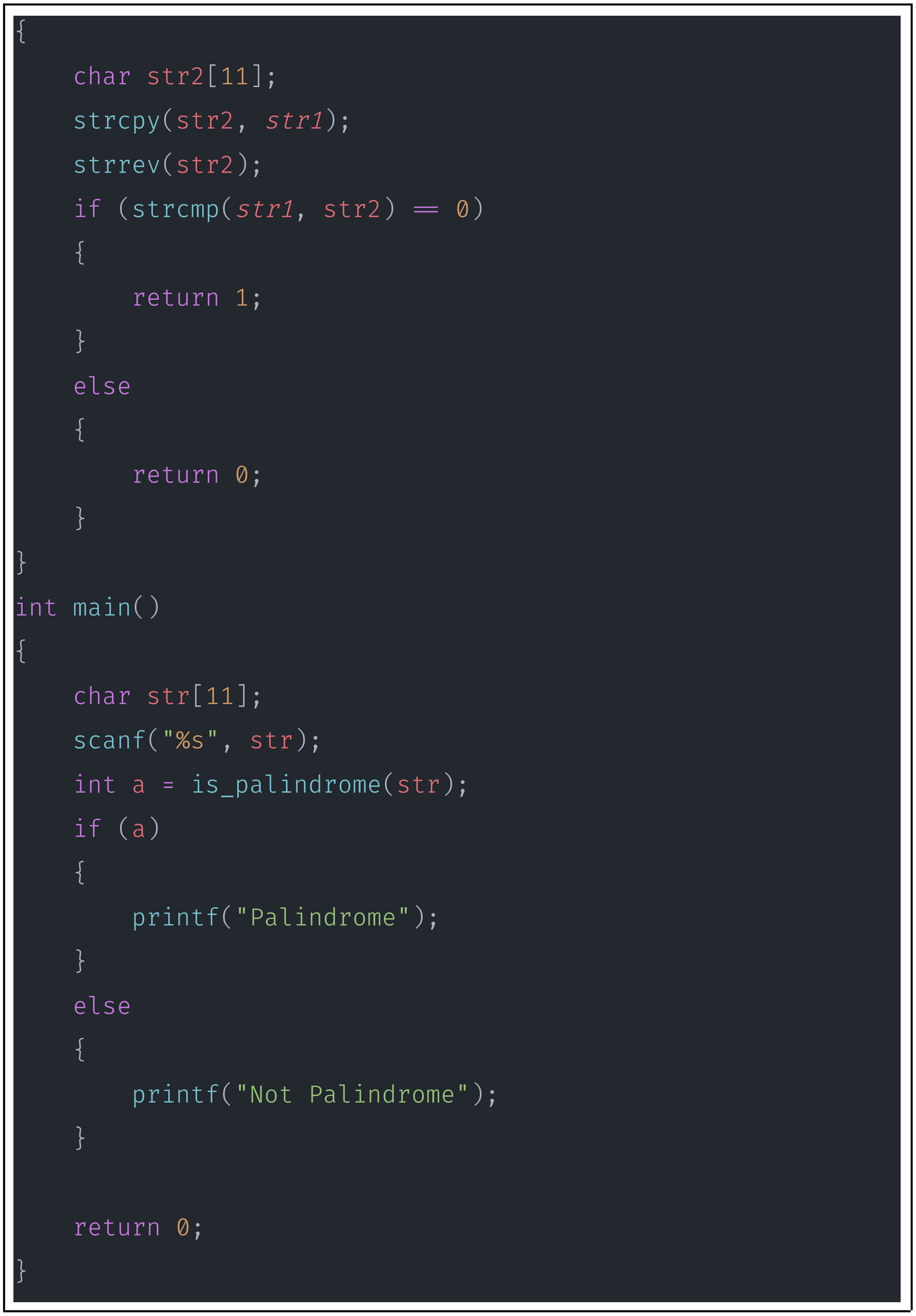
```c
    printf("Hello World!\n");

}



int main()

{

    greet(); // prints "Hello World!"

    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

```c
#include <stdio.h>
#include <string.h>
int is_palindrome(char *str1)
```



```c
{
    char str2[11];
    strcpy(str2, str1);
    strrev(str2);
    if (strcmp(str1, str2) == 0)
    {
        return 1;
    }
    else
    {
        return 0;
    }
}
int main()
{
    char str[11];
    scanf("%s", str);
    int a = is_palindrome(str);
    if (a)
    {
        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 means that a copy of the argument is passed to the function. Any changes made to the argument inside the function do not affect the original variable outside the function. Here's an example:

```c
#include <stdio.h>

void increment(int x)
{
    x++;
}

int main()
{
    int num = 20;
    increment(num);
    printf("num = %d\n", num); // Output: num = 20
    return 0;
}
```

On the other hand, pass by reference means that the memory address of the argument is passed to the function. This allows the function to modify the original variable directly. Here's an example:

```c
#include <stdio.h>

void increment(int *x)
{
    (*x)++;
}

int main()
{
    int num = 20;
    increment_(&num);
    printf("num = %d\n", num); // Output: num = 21
    return 0;
}
```