Ashutosh Shrivastava

SAP ID – 590027859

Batch 78

1. Function: printf()

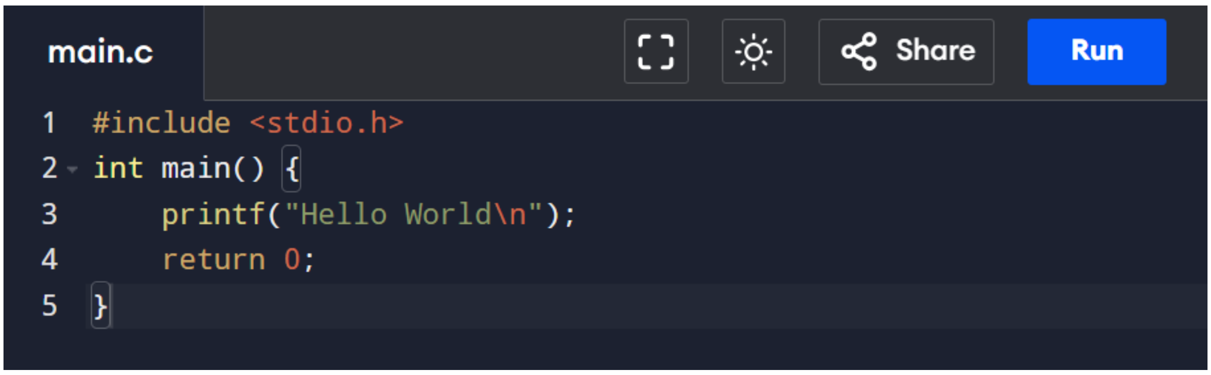
Header: stdio.h

Algorithm

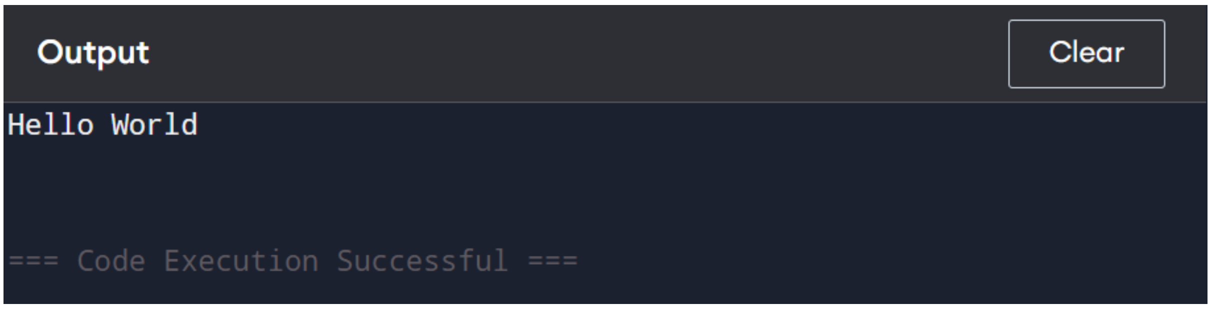
1. Start
2. Use printf() to display a message
3. End

Pseudocode

```
START
PRINT “Hello World”
END
```



```c
#include <stdio.h>
int main() {
    printf("Hello World\n");
    return 0;
}
```



```
Hello World


=== Code Execution Successful ===
```

2. Function: scanf()

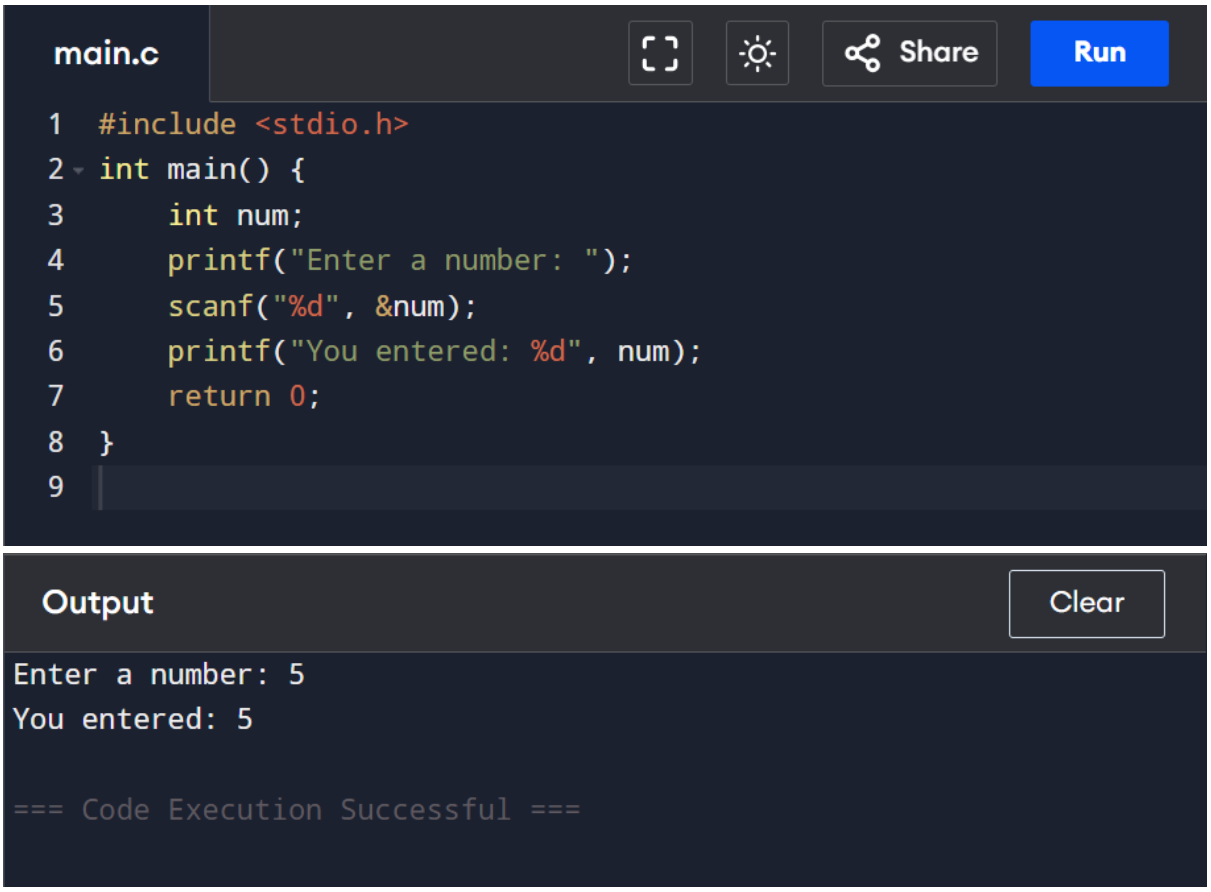
Header: stdio.h

Algorithm

1. Start
2. Ask user for a number
3. Read number using scanf()
4. Print number
5. End

Pseudocode

```
START
INPUT num
PRINT num
END
```



```c
#include <stdio.h>
int main() {
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);
    printf("You entered: %d", num);
    return 0;
}
```

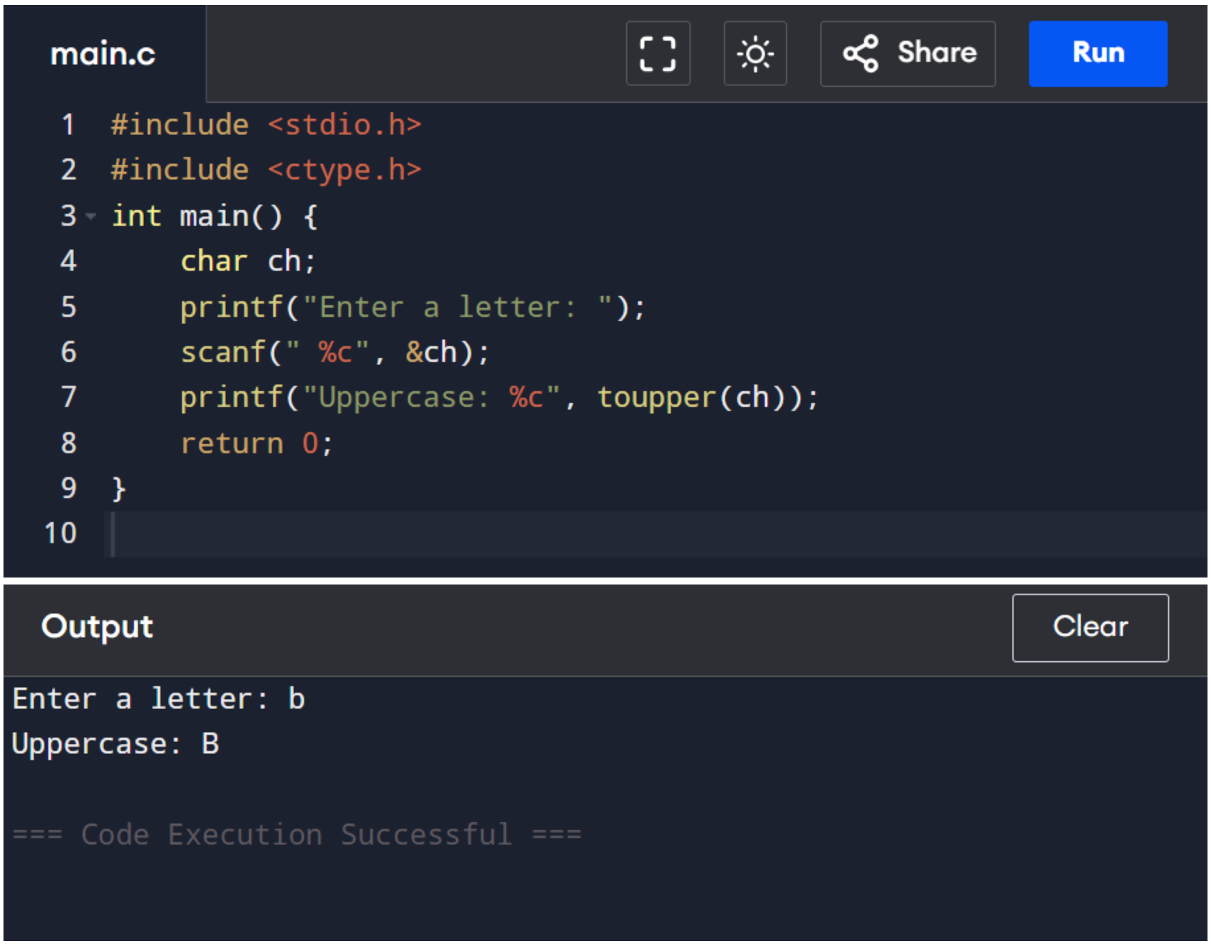
```
Enter a number: 5
You entered: 5

=== Code Execution Successful ===
```

3. Function: toupper()

Header: ctype.h

Algorithm

1. Start
2. Input a character
3. Convert to uppercase using toupper()
4. Print result
5. End

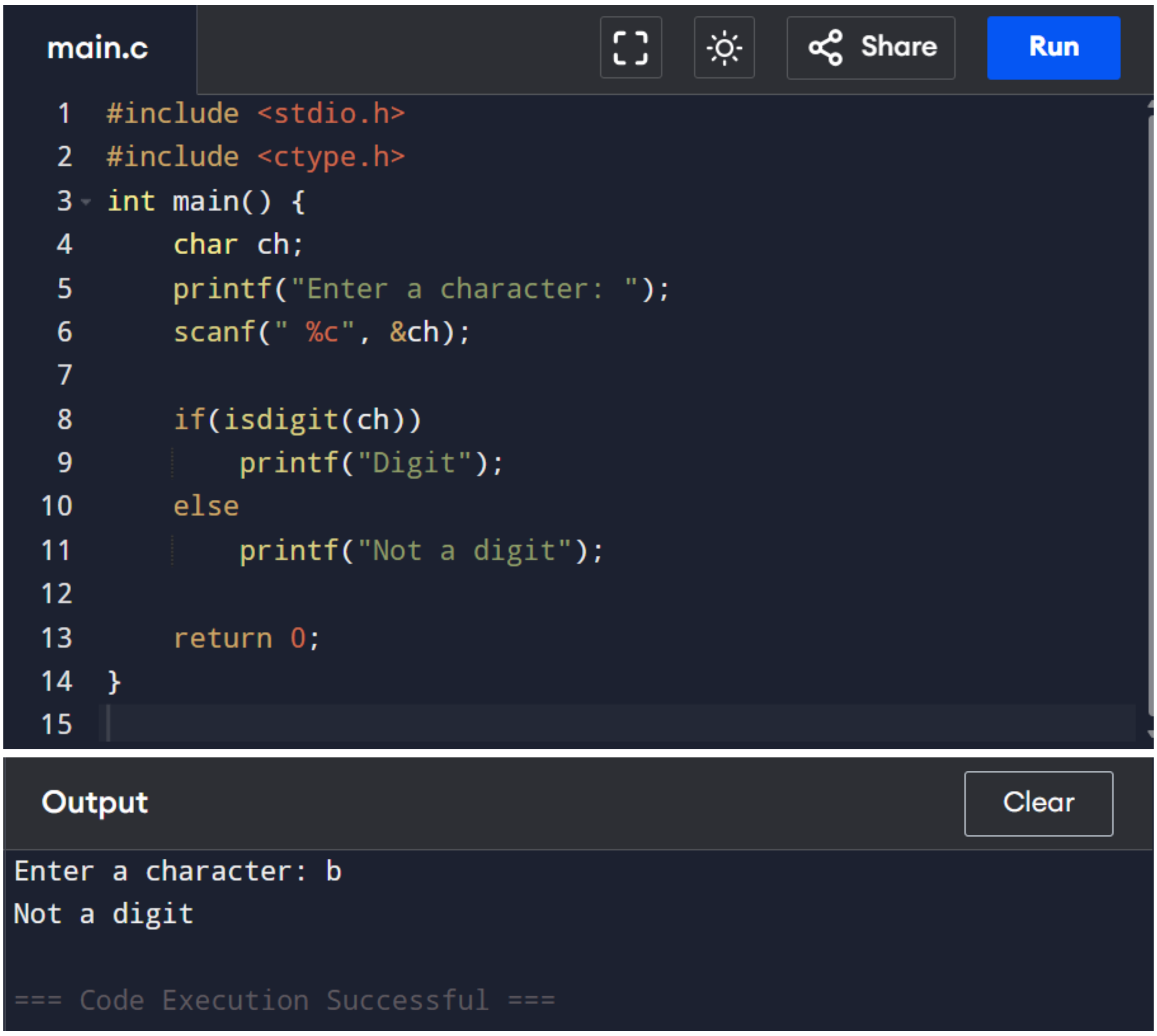


```c
#include <stdio.h>
#include <ctype.h>
int main() {
    char ch;
    printf("Enter a letter: ");
    scanf(" %c", &ch);
    printf("Uppercase: %c", toupper(ch));
    return 0;
}
```

```
Enter a letter: b
Uppercase: B

=== Code Execution Successful ===
```

4. Function: isdigit()

Header: ctype.h

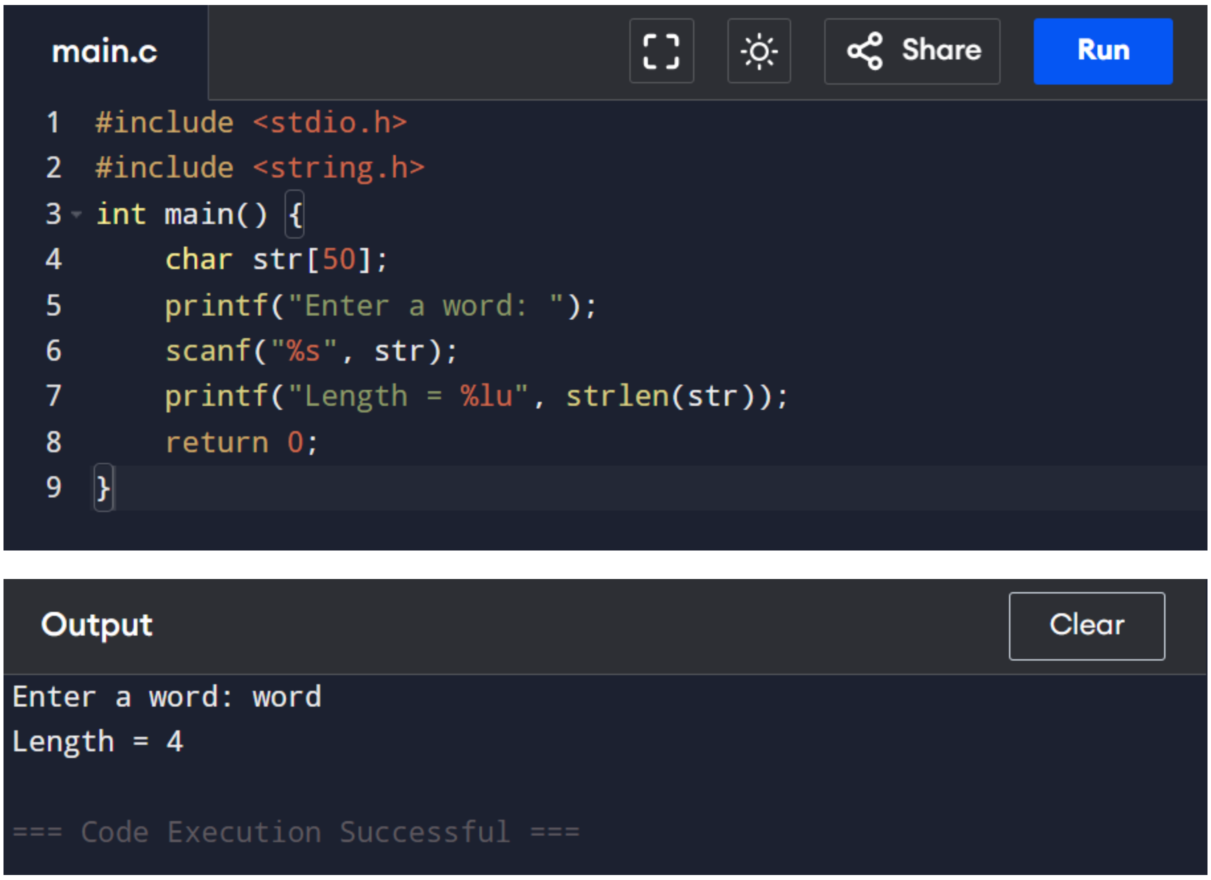


```c
#include <stdio.h>
#include <ctype.h>
int main() {
    char ch;
    printf("Enter a character: ");
    scanf(" %c", &ch);

    if(isdigit(ch))
        printf("Digit");
    else
        printf("Not a digit");

    return 0;
}

```

```
Enter a character: b
Not a digit

=== Code Execution Successful ===
```

5. Function: strlen()

Header: string.h

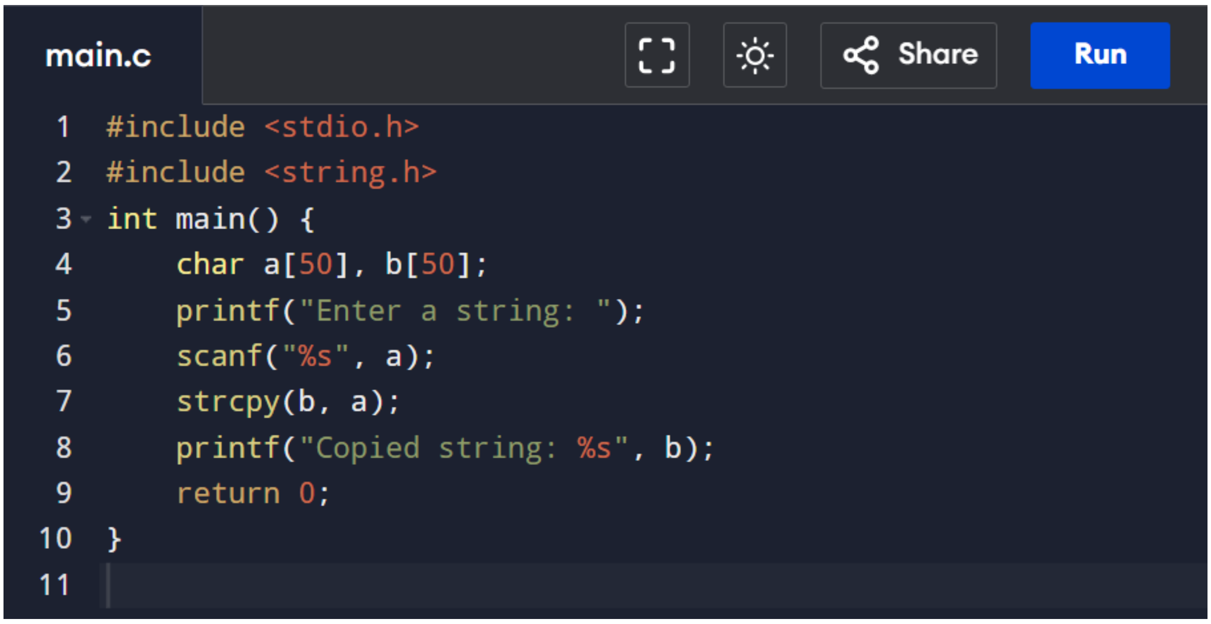


```c
#include <stdio.h>
#include <string.h>
int main() {
    char str[50];
    printf("Enter a word: ");
    scanf("%s", str);
    printf("Length = %lu", strlen(str));
    return 0;
}
```

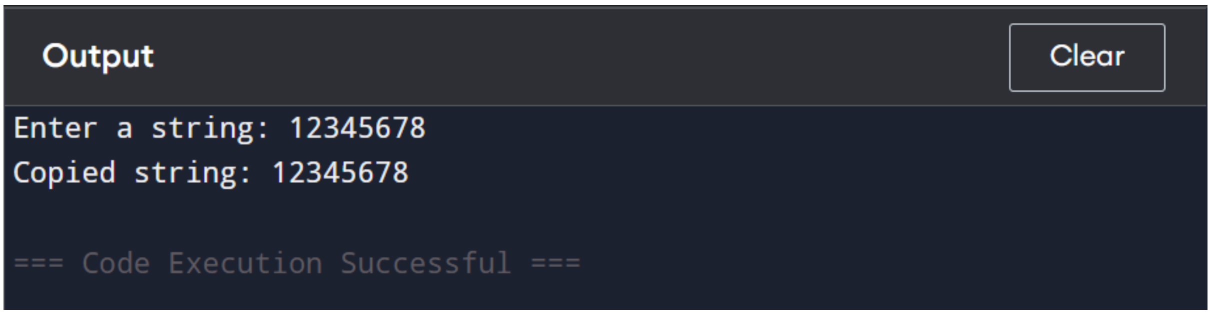
```
Enter a word: word
Length = 4

=== Code Execution Successful ===
```

6. Function: strcpy()

Header: string.h



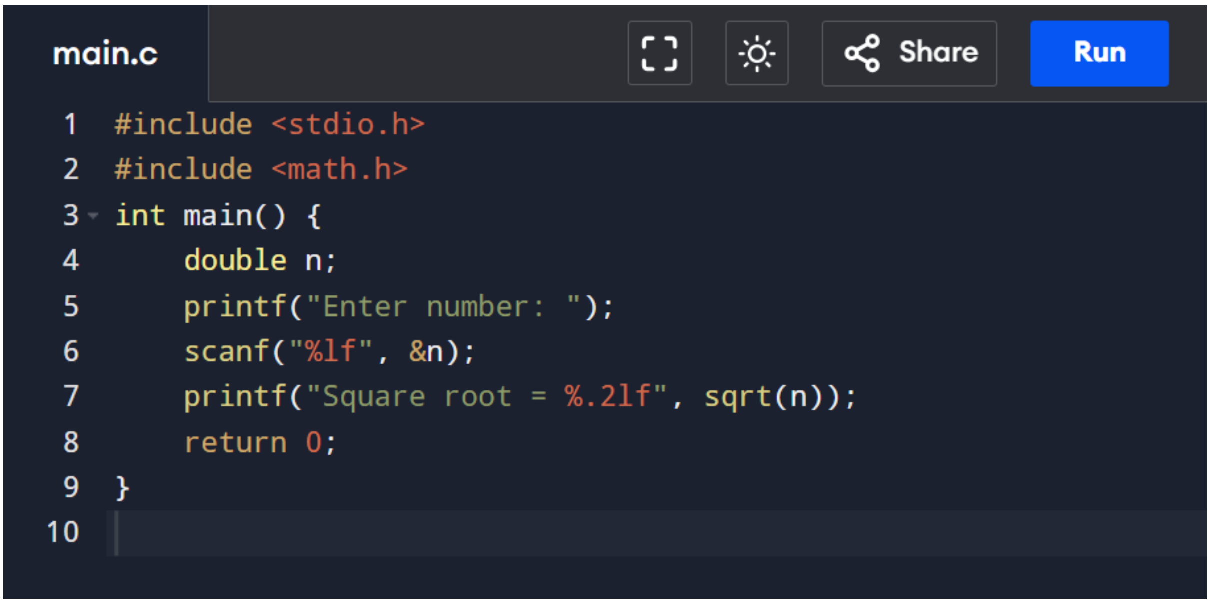
```c
#include <stdio.h>
#include <string.h>
int main() {
    char a[50], b[50];
    printf("Enter a string: ");
    scanf("%s", a);
    strcpy(b, a);
    printf("Copied string: %s", b);
    return 0;
}
```



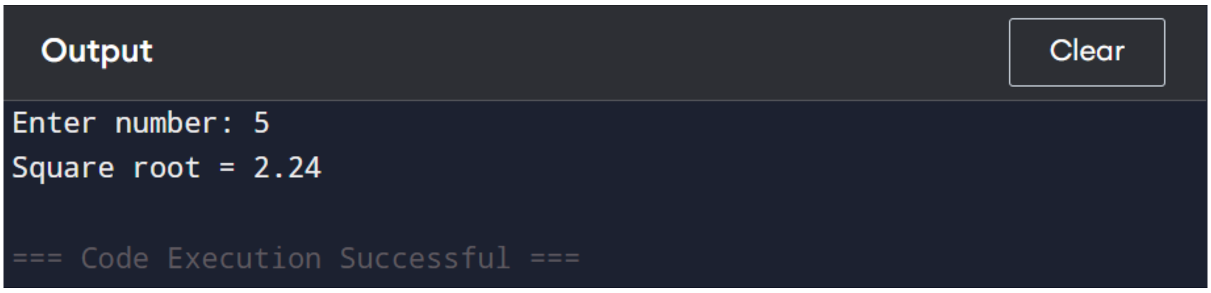
```
Enter a string: 12345678
Copied string: 12345678

=== Code Execution Successful ===
```

7. Function: sqrt()

Header: math.h



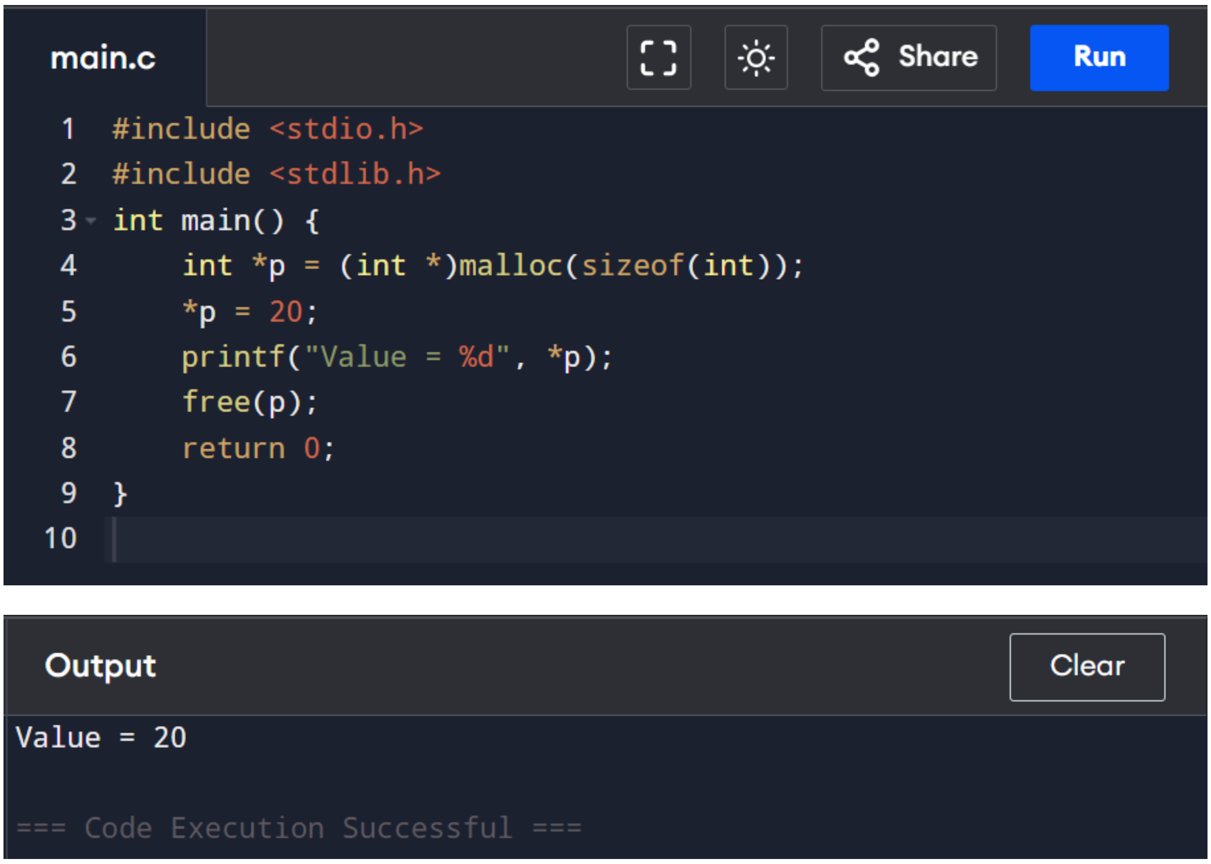
```c
#include <stdio.h>
#include <math.h>
int main() {
    double n;
    printf("Enter number: ");
    scanf("%lf", &n);
    printf("Square root = %.2lf", sqrt(n));
    return 0;
}
```



```
Enter number: 5
Square root = 2.24

=== Code Execution Successful ===
```

8. Function: malloc()

Header: stdlib.h



```c
#include <stdio.h>
#include <stdlib.h>
int main() {
    int *p = (int *)malloc(sizeof(int));
    *p = 20;
    printf("Value = %d", *p);
    free(p);
    return 0;
}
```

```
Value = 20


=== Code Execution Successful ===
```

9. Function: free()

Header: stdlib.h

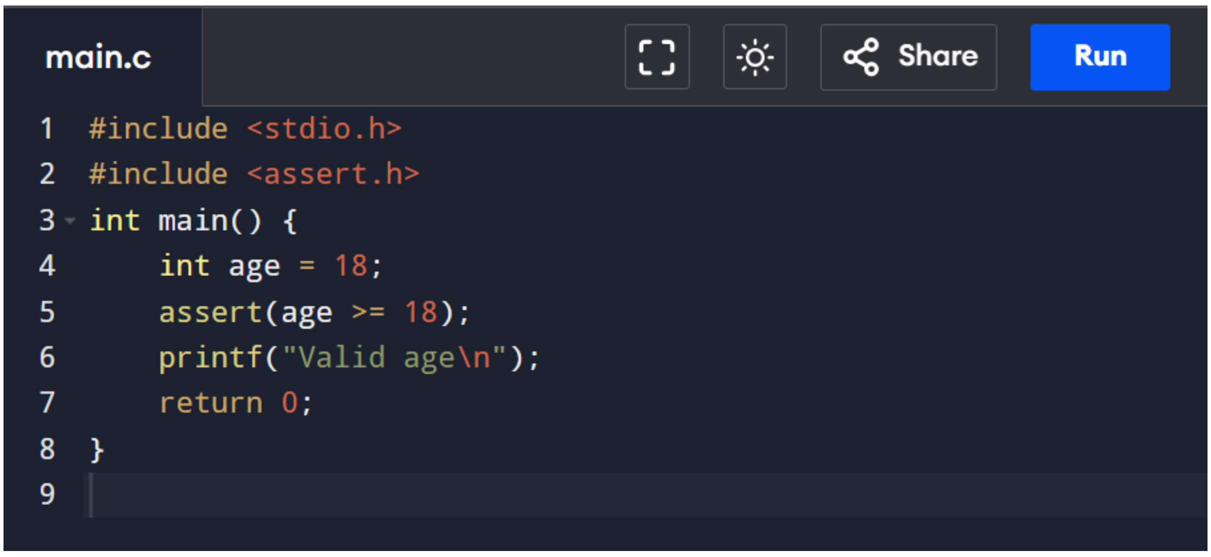
(Used together with malloc)

Explanation:

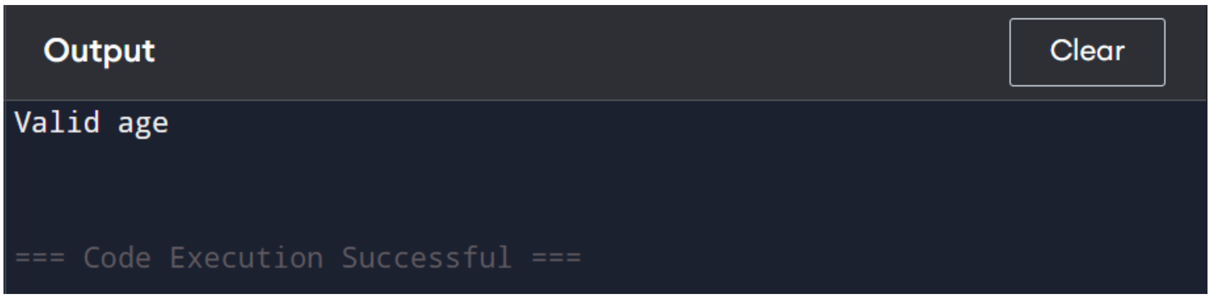
free(ptr) releases memory that was allocated using malloc().

10. Function: assert()

Header: assert.h



```c
#include <stdio.h>
#include <assert.h>
int main() {
    int age = 18;
    assert(age >= 18);
    printf("Valid age\n");
    return 0;
}

```



```
Valid age


=== Code Execution Successful ===
```

Q1. Static Library vs Shared Library

Static Library (.a / .lib)

- Added to the program during compilation
- Final executable becomes bigger
- Faster execution
- No need for library file during runtime

Shared Library (.so / .dll)

- Linked during program execution
- Reduces executable size
- Same library can be used by many programs
- Requires library file during runtime

Q2. **Dynamic Memory vs Static Memory**

Static Memory

- Allocated at compile time
- Fixed size
- Example:

```
int a[10];
```

Dynamic Memory

- Allocated at runtime
- Flexible size
- Uses malloc(), calloc(), realloc()
- Must use free() to release memory

Example:

```
int *p = malloc(5 * sizeof(int));
```

Q3. **Garbage Collection**

- C **does not have automatic garbage collection**
- Programmer must manually free memory
- Use free() for memory allocated by malloc()
- Prevents memory leaks