1. Write a C program that takes an integer input and multiplies it by 2^n using the left shift operator.

#include<stdio.h>

int main()

{

int num,exp;

printf("Enter the number: ");

scanf("%d", &num);

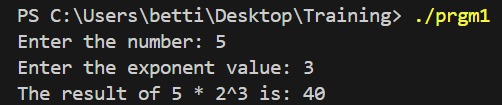
printf("Enter the exponent value: ");

scanf("%d", &exp);

printf("The result of %d \* 2^%d is: %d",num,exp,num\*(1<<exp));

return 0;

}



2. Create a C program that counts how many times you can left shift a number before it overflows (exceeds the maximum value for an integer).

#include<stdio.h>

int main()

{

int num, value, flag =0;

printf("Enter the number: ");

scanf("%d", &num);

value = num;

while (value <= 2147483647)

{

if (value < 0) {

break;

}

value = value << 1;

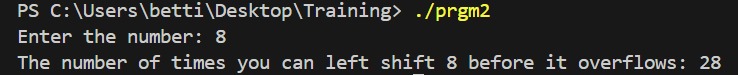
flag+=1;

}

printf("The number of times you can left shift %d before it overflows: %d\n", num, flag);

return 0;

}



3. Write a C program that creates a bitmask with the first n bits set to 1 using the left shift operator.

#include<stdio.h>

int main()

{

int n,number=0;

printf("Enter the number of bits to set 1: ");

scanf("%d", &n);

for(int i = 0; i < n; i++)

{

int mask = 1 << i;

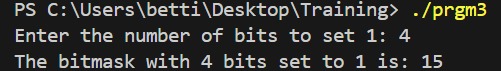
number = mask | number;

}

printf("The bitmask with %d bits set to 1 is: %d\n", n, number);

return 0;

}



4. Develop a C program that reverses the bits of an integer using left shift and right shift operations.

#include <stdio.h>

int main()

{

int num = 0, rev = 0;

printf("Enter the number: ");

scanf("%d", &num);

for (int i = 0; i < 8; i++)

{

rev = (rev << 1) | (num & 1);

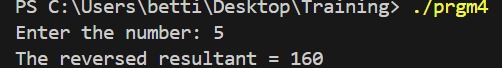
num = num >> 1;

}

printf("The reversed resultant = %d\n", rev);

return 0;

}



5. Create a C program that performs a circular left shift on an integer.

#include<stdio.h>

int main()

{

unsigned char num;

printf("enter num: ");

scanf("%hhu", &num);

unsigned char num1 = (num << 1) | (num >> 7);

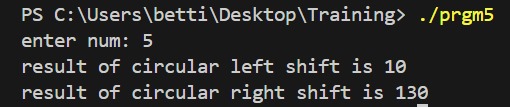
unsigned char num2 = (num >> 1) | (num << 7);

printf("result of circular left shift is %u\n", num1);

printf("result of circular right shift is %u\n", num2);

return 0;

}



6. Write a C program that takes an integer input and divides it by 2^ n using the right shift operator.

#include<stdio.h>

int main()

{

int num, exp;

printf("Enter the number: ");

scanf("%d", &num);

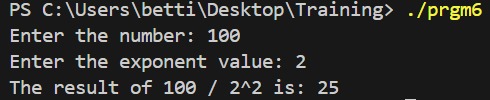
printf("Enter the exponent value: ");

scanf("%d", &exp);

printf("The result of %d / 2^%d is: %d", num, exp, num >> exp);

return 0;

}



7. Create a C program that counts how many times you can right shift a number before it becomes zero.

#include<stdio.h>

int main()

{

int num, value, flag =0;

printf("Enter the number: ");

scanf("%d", &num);

value = num;

while (value > 0)

{

value = value >> 1;

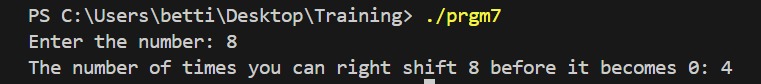
flag+=1;

}

printf("The number of times you can right shift %d before it becomes 0: %d\n", num, flag);

return 0;

}



8. Write a C program that extracts the last n bits from a given integer using the right shift operator.

#include <stdio.h>

#include <stdint.h>

int main() {

uint16\_t num, n;

printf("Enter the number: ");

scanf("%hd", &num);

printf("Enter the number of last bits to be extracted from the number: ");

scanf("%hd", &n);

uint16\_t mask = (1 << n) - 1;

uint16\_t extracted\_bits = num & mask;

printf("Binary of the number: ");

for (int i = 15; i >= 0; i--) {

printf("%d", (num >> i) & 1);

}

printf("\n");

printf("The last %d bits of the number %d are: %d\n", n, num, extracted\_bits);

printf("Binary of the extracted bits: ");

for (int i = n - 1; i >= 0; i--) {

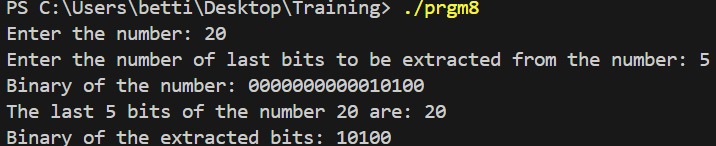
printf("%d", (extracted\_bits >> i) & 1);

}

printf("\n");

return 0;

}



9. Develop a C program that uses the right shift operator to create a bitmask that checks if specific bits are set in an integer.

#include <stdio.h>

#include <stdint.h>

int main() {

uint16\_t num, mask;

int n;

int check\_bits[10];

printf("Enter the number: ");

scanf("%hd", &num);

printf("How many bits are to be checked: ");

scanf("%d", &n);

printf("Enter the positions of the bits to be checked: ");

for (int i = 0; i < n; i++) {

scanf("%d", &check\_bits[i]);

}

for (int i = 0; i < n; i++) {

mask = mask|(1 << check\_bits[i]);

}

for (int i = 0; i < n; i++) {

if (num & mask) {

printf("Bit %d is set (1).\n", check\_bits[i]);

} else {

printf("Bit %d is not set (0).\n", check\_bits[i]);

}

}

return 0;

}

