```c
int chosen;
char temp[20];
int sum;

int ask(){
    printf("Choose a number");
    scanf("%d", &chosen);

    int i;
    for (i = 0; i <= chosen; i ++){
        sum += i;
    }

    sprintf(temp, "%d", sum);

    return 0;

}
```

- This function calculates the sum as stated by the task using a simple for loop. It will be called later in the “main” function of the code.
- The “temp” variable is used to convert the “sum” variable to a string so that it could be stored in a *const char data type variable.
- Interesting discussion point of this code is that if the string“temp” did not have any initial specified size, the program would produce a “Segmentation Fault” error which means that there is a memory problem.
- This function is a part of the producer.c (child process code)

```c
int main(){
    const int SIZE = 8192;
    const char *name = "OS";

    const char *message_0 = temp;
    const char *message_1 = " is the answer";
    int shm_fd;
    void *ptr;

    ask();

    shm_fd = shm_open(name, O_CREAT | O_RDWR, 0666);
    ftruncate(shm_fd, SIZE);
    ptr = mmap(0, SIZE, PROT_WRITE, MAP_SHARED, shm_fd, 0);

    sprintf(ptr, "%s", message_0);
    ptr += strlen(message_0);
    sprintf(ptr, "%s", message_1);
    ptr += strlen(message_1);

    return 0;
}
```

- This function is the main function of the “producer.c” file. This code is very similar to the example code which writes two strings (“hello” and “world”) to the shared memory. Instead of storing those strings (defined as message_0 and message_1 in the code as constant strings), it stores the result of the for loop defined by the “ask” function (converted as a string) and the string “is the answer”. The shared memory size is also twice as much compared to the example code (just for testing purposes).
- The “shm_open” creates the shared memory object, the “ftruncate” changes the size of the shared memory object.
- The mmap function which utilizes a pointer by storing its returned value is used to map the memory object by using the variables created and mentioned above.
- The “sprintf” section of the code is used to write the strings to the memory.

```c
# include <fcntl.h>
# include <sys/shm.h>
# include <sys/stat.h>
# include <sys/mman.h>

int main(){
    const int SIZE = 8192;
    const char *name = "OS";
    int shm_fd;
    void *ptr;

    shm_fd = shm_open(name, O_RDONLY, 0666);
    ptr = mmap(0, SIZE, PROT_READ, MAP_SHARED, shm_fd, 0);
    printf("%s", (char *)ptr);
    shm_unlink(name);

    return 0;
}
```

- This function is the only function in the consumer.c (parent process) side. This code is exactly the same as the one shown inside the example except for the size of the memory specified is twice as much (have to be the same as the producer side).
- The “shm_open” function is different from the producer part slightly since the memory is opened as a “read only” state. Similar procedure as the “mmap” function’s argument.
- This code reads the shared memory that was written from the producer side and then after printing out the strings in the memory, it removes the shared memory object.