Operation System HW1 **b03705002 林軒逸**

Code for Problem 1:

```c
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
int main()
{

    int*fib; //pointer of dynamic array of fibonacci sequence
    int numbers;// the number user inputted
    scanf("%d",&numbers); //read the number
    while (numbers<0) { //error detecting of negative number
        printf("Please input a positive number:");
        scanf("%d",&numbers);
    }
    fib= (int*)malloc(numbers*sizeof(int)); //declaration of the dynamic array
    pid_t pid;
    pid = fork(); /* fork another process */
    if (pid < 0) { /* error occurred */
        printf("Fork Failed\n");
        exit(-1);
    }
    else if (pid == 0) { /* child process */
        for(int i=0;i<numbers;i++) //fibnocci sequence
        {
            if(i==0)
            {
                fib[i]=0;
                printf("%d ",fib[i]); //output
            }
            else if(i==1)
            {
                fib[i]=1;
                printf("%d ",fib[i]); //output
            }
            else
            {
                fib[i]=fib[i-1]+fib[i-2];
                printf("%d ",fib[i]); //output
            }
        }
    }
    else { /* parent process */
        /* parent will wait for the child to complete */
        wait(NULL);
        printf ("Child Complete\n");
        exit(0);
    }
}
```

Explain for Problem 1:

First, I create a dynamic array call *fib* to create space for storing the Fibonacci sequence with length *numbers*.

numbers is for storing the number the user entered, which is the length of fib.

After that, we use the function *fork()* to create process. In the child process, we create the Fibonacci sequence and print all the required values out, and exit at the parent process.

Code for Problem 2:

```
#include <sys/types.h>
#include <stdio.h>
#include <sys/shm.h>
#include <sys/ipc.h>
#include <sys/stat.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>

#define MAX_SEQUENCE 10
typedef struct{
    long fib_sequence[MAX_SEQUENCE];
    int sequence_size;
}shared_data;

int main() {
    int numbers; //the output numbers
    scanf("%d",&numbers);
    //Ensuring number>0
    while (numbers<0) {
        printf("Please enter a positive number.\nPlease enter:");
        scanf("%d",&numbers);
    }
    //Ensuring number<= MAX_SEQUENCE
    while(numbers>MAX_SEQUENCE)
    {
        printf("Please enter a number smaller than MAX_SEQUENCE!\nPlease enter:");
        scanf("%d",&numbers);
        while (numbers<=0) //Ensuring number>0
        {
            printf("Please enter a positive number.\nPlease enter:");
            scanf("%d",&numbers);
        }
    }
    // create share memory
    int segment_id;
    shared_data *shared_memory_fib; //The share memory pointer
    int size= sizeof(shared_memory_fib->sequence_size);
    segment_id = shmget(IPC_PRIVATE, size*MAX_SEQUENCE, S_IRUSR | S_IWUSR);
    // attach the shared memory segment
    shared_memory_fib = (shared_data*) shmat(segment_id,NULL,0);
    memset(shared_memory_fib,0,size);
        shared_memory_fib->sequence_size=numbers;
    // create process
    pid_t pid;
    pid = fork();
    if (pid < 0) {
        fprintf(stderr, "Fork Failed"); exit(-1);
    }


//child process, creating fibonacci sequence
else if (pid == 0) {
    //Creating fibonacci sequence
    for(int i=0;i<shared_memory_fib->sequence_size;i++)
    {
        if(i==0)
            shared_memory_fib->fib_sequence[i]=0;
        else if(i==1)
            shared_memory_fib->fib_sequence[i]=1;
        else
            shared_memory_fib->fib_sequence[i]=shared_memory_fib->fib_sequence[i-1]+shared_memory_fib->fib_sequence[i-2];
    }
    exit(0);
}
//parent process, outputting fibonaaci sequence
else {
    int count = 1;
    while (count > 0) {
        wait(NULL);
        count--;
        if(shared_memory_fib->sequence_size>MAX_SEQUENCE)
            //error checking, although we had do it at the very first
        {
            exit(0);
        }
        //outputing fibonacci sequence
        for(int i=0;i<shared_memory_fib->sequence_size;i++)
        {
            printf("%ld ",shared_memory_fib->fib_sequence[i]);
        }
        // detach the segment then exit
        shmdt(shared_memory_fib);
        printf("\nProgram executed.");
        exit(0);
    }
}
}
```

Explain of Problem 2:

First, we use the structure that the problem described, and create a pointer calls *shared_memory_fib* to do all the process.

Second, we check whether the input number is less or equal to 0 or bigger than MAX_SEQUENCE, we will ask the user to re-input a valid number.

After that, we create share memory by using *shared_memory_fib* and some functions, and then use *fork()* to create processes.

In child function, we calculate the Fibonacci sequence with length *sequence_size*, and in parent function, we output the Fibonacci sequence, and detach with shared memory, and then exit the program.

That's all for my HW1, thanks for reading!