THREADS

NAME	RAVELLA ABHINAV
ROLL No.	CB.EN.U4CSE19453

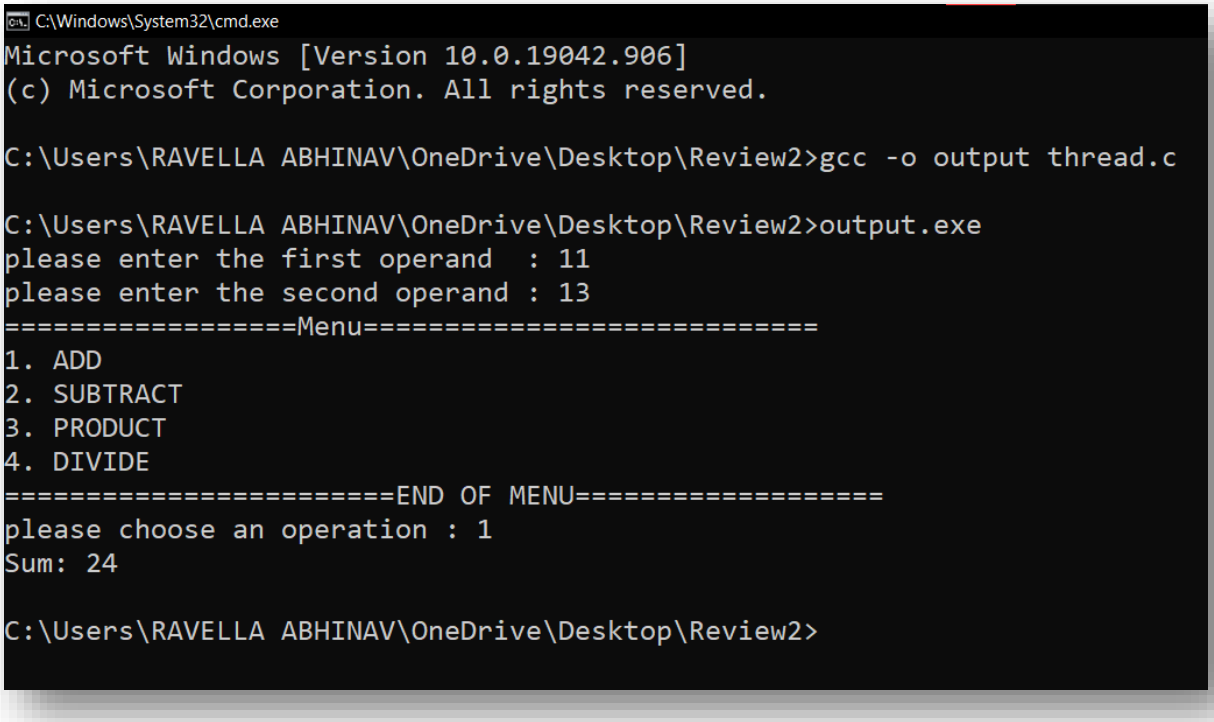
```
#include <windows.h>
#include <stdio.h>

DWORD Sum_of;
DWORD Diff;
DWORD product;
DWORD divi;

typedef struct parameters{
    int operand1;
    int operand2;
}param, *para;


DWORD WINAPI Sum(LPVOID Param){

    para Upper = (para)Param;

    Sum_of = Upper->operand1 + Upper->operand2;

    return 0;
}


DWORD WINAPI Difference(LPVOID Param){
    para Upper = (para)Param;

    if (Upper->operand1 > Upper->operand2){
        Diff = Upper->operand1 - Upper->operand2;
    }
    else{
        Diff = Upper->operand2 - Upper->operand1;
    }
    return 0;
}


DWORD WINAPI Product(LPVOID Param){
    para Upper = (para)Param;
    product = Upper->operand1 * Upper->operand2;
    return 0;
}


DWORD WINAPI Divide(LPVOID Param){
```

```
    para Upper = (para)Param;

    if (Upper->operand2 == 0){
        divi = -1;
    }
    else{
        divi = Upper->operand1/Upper->operand2;
    }
    return 0;
}


int main(){
    DWORD thread_id;
    HANDLE ThreadHandle;
    HANDLE ThreadHandle1;
    HANDLE ThreadHandle2;
    HANDLE ThreadHandle3;

    struct parameters param;
    struct parameters *params;

    params = &param;

    printf("please enter the first operand  : ");
    scanf("%d",&params->operand1);

    printf("please enter the second operand : ");
    scanf("%d",&params->operand2);

    int operation;

    printf("==================Menu============================\n");
    printf("1. ADD\n");
    printf("2. SUBTRACT\n");
    printf("3. PRODUCT\n");
    printf("4. DIVIDE\n");
    printf("========================END OF MENU==================\n");

    printf("please choose an operation : ");
    scanf("%d",&operation);

    if (operation == 1){

        ThreadHandle = CreateThread(
                NULL,     //default security attribute
                0,        //default stack size of the thread
                Sum,   //function that is called by the thread
                params,      //structure of arguments that are passed into
the threads
                0,         //default creation flags
                &thread_id     // the thread returns the thread id here
        );

        //this system call will wait for the thread to finish, for infinite
amount of time.

        WaitForSingleObject(ThreadHandle,INFINITE);

        //closing the thread here
```

```
        CloseHandle(ThreadHandle);

        printf("Sum: %lu\n",Sum_of);
    }

    else if(operation == 2){

        ThreadHandle1 = CreateThread(
                NULL,      //default security attribute
                0,         //default stack size of the thread
                Difference,    //function that is called by the thread
                params,      //structure of arguments that are passed into
the threads
                0,          //default creation flags
                &thread_id      // the thread returns the thread id here
        );

        //this system call will wait for the thread to finish, for infinite
amount of time.

        WaitForSingleObject(ThreadHandle1,INFINITE);

        //closing the thread here

        CloseHandle(ThreadHandle1);

        printf("Difference: %lu\n",Diff);
    }

    else if(operation == 3){
        ThreadHandle2 = CreateThread(
                NULL,      //default security attribute
                0,         //default stack size of the thread
                Product,    //function that is called by the thread
                params,      //structure of arguments that are passed into
the threads
                0,          //default creation flags
                &thread_id      // the thread returns the thread id here
        );

        //this system call will wait for the thread to finish, for infinite
amount of time.

        WaitForSingleObject(ThreadHandle2,INFINITE);

        //closing the thread here

        CloseHandle(ThreadHandle2);

        printf("product: %lu\n",product);
    }

    else{
        ThreadHandle3 = CreateThread(
                NULL,      //default security attribute
                0,         //default stack size of the thread
                Divide,    //function that is called by the thread
                params,      //structure of arguments that are passed into
the threads
                0,          //default creation flags
                &thread_id      // the thread returns the thread id here
```

```
        );

        //this system call will wait for the thread to finish, for infinite
amount of time.

        WaitForSingleObject(ThreadHandle3,INFINITE);

        //closing the thread here

        CloseHandle(ThreadHandle3);

        printf("Quotient: %lu\n",divi);
    }

}
```

Output :



```
C:\Users\RAVELLA ABHINAV\OneDrive\Desktop\Review2>gcc -o output thread.c

C:\Users\RAVELLA ABHINAV\OneDrive\Desktop\Review2>output.exe
please enter the first operand  : 11
please enter the second operand : 13
==================Menu============================
1. ADD
2. SUBTRACT
3. PRODUCT
4. DIVIDE
========================END OF MENU==================
please choose an operation : 2
Difference: 2

C:\Users\RAVELLA ABHINAV\OneDrive\Desktop\Review2>
```

```
C:\Users\RAVELLA ABHINAV\OneDrive\Desktop\Review2>output.exe
please enter the first operand  : 11
please enter the second operand : 13
==================Menu============================
1. ADD
2. SUBTRACT
3. PRODUCT
4. DIVIDE
========================END OF MENU==================
please choose an operation : 3
product: 143

C:\Users\RAVELLA ABHINAV\OneDrive\Desktop\Review2>output.exe
please enter the first operand  : 11
please enter the second operand : 13
==================Menu============================
1. ADD
2. SUBTRACT
3. PRODUCT
4. DIVIDE
========================END OF MENU==================
please choose an operation : 4
Quotient: 0

C:\Users\RAVELLA ABHINAV\OneDrive\Desktop\Review2>
```