Independent University Bangladesh

Programming Project

MAHIDI HASAN MITHUN

ID: 1930432

Semester: Spring 2023

Section: 02

Operating System

Course ID: CSE315

Instructor: Mohammad Noor Nobi

Ans to the question – A:

```c
#include <stdio.h>
#include<stdio.h>
#include<sys/wait.h>
#include<unistd.h>
#include<stdlib.h>

int main(){
    int n;
    int process;

    printf("Enter Any Number: ");
    scanf("%d",&n);

    int fork();

    // fork() function call returns process id, stored in pid
    // with posibility of being positive, negative or zero
    process = fork();

    if (process < 0){ //program enters here if child process is not created
        printf("Failed to Create Child\n");
        exit(0);
    }

    else if ( process == 0 ) { //Child Process
        if(n > 0) { //check if positive integer is entered
            while(n > 0){
                printf("%d ",n); //prints the sequence
                if(n == 1){
                    break; //breaks if n equals to 1
                }

                else if(n%2 == 0){
                    n = n / 2; //n value changes if n is even
                }

                else{
                    n = (3 * n) + 1; //n value changes if n is odd
                }
            }
        }
        else {
            printf("\nPlease Enter a Positive Integer");
        }
        exit(0); //terminates child process
    }

    else { // parent process
        // printf("\nWaiting for Child process to end...");
        wait(NULL); // waiting for child process to end
    }

    return 0;
}
```

Ans to the question – B:

```c
#include <sys/ipc.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>

int main() {
    int n = 3;
    int fd[2 * n];
    char write_msg[n][100];
    char read_msg[100];
    int pid;
    for (int i = 0; i < n; i++) {
        if (pipe(&fd[2 * i]) == -1) {
            printf("Error creating pipe\n");
            return -1;
        }
    }
    for (int i = 0; i < n; i++) {
        if (pid = fork() == 0) {
            printf("Writing to Child Process %d\n",i);
            close(fd[2 * i]);
            int j = 0;
            int counter = 0;

            while (1) {
                char in;
                scanf("%c", &in);

                if (in == '\n') {
                    counter++;
                    if (counter == 2) {
                        break;
                    }
                }
                else {
                    write_msg[i][j] = in;
                    j++;
                }
            }
            write_msg[i][j] = '\0';
            write(fd[2 * i + 1], &write_msg[i], sizeof(write_msg[i]));
            close(fd[2 * i + 1]);
            exit(0);
        }
        else {
            wait(&pid);
        }
    }
    for (int i = 0; i < n; i++) {
        read(fd[2 * i], read_msg, sizeof(read_msg));
        close(fd[2 * i]);
        printf("\nReading from child process %d of parent: %s", i, read_msg);
    }
    printf("\n");
    return 0;
}
```

Ans to the question – C:

```c
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

#define SIZE 10
#define NUMBER_OF_THREADS 3
void *sorter(void *params); // thread that performs basic sorting algorithm
void *merger(void *params); //thread that performs merging of results

int list[SIZE] = {7,1,8,3,4,2,6,9,12,15};
int result[SIZE];

typedef struct {
    int from_index;
    int to_index;
} parameters;



//sorting function
void *sorter(void *params){
    parameters* p = (parameters *)params;

    int begin = p->from_index;
    int end = p->to_index+1;
    int z;
    int i,j,t,k;

    for(i=begin; i< end; i++){
        for(j=begin; j< end-i-1; j++){
            if(list[j] > list[j+1]){
                t = list[j];
                list[j] = list[j+1];
                list[j+1] = t;
            }
        }
    }
    int x;
    for(x=begin; x<end; x++){
        result[x]=list[x] ;
    }
    pthread_exit(0);
}

//merge function
void *merger(void *params){
    parameters* p = (parameters *)params;

    int begin = p->from_index;
    int end = p->to_index+1;
    int i,j,t;

    for(i=begin; i< end; i++){
        for(j=begin; j< end-i; j++){
            if(result[j] > result[j+1]){
                t = result[j];
                result[j] = result[j+1];
                result[j+1] = t;
            }
        }
    }
    int d;
    printf("The Final Resulting Array is: ");
    for(d=0; d<SIZE; d++){
        printf(" %d ", result[d]);
    }
    printf("\n");
    pthread_exit(0);
}


int main (int argc, const char * argv[]) {
    int i;
    printf("The Unsorted Array: ");
    for (i = 0; i<SIZE; ++i){
        printf(" %d ", list[i]);
    }
    printf("\n");

    pthread_t workers[NUMBER_OF_THREADS];
    //establish the first sorting thread

    parameters *data = (parameters *) malloc (sizeof(parameters));
    data->from_index = 0;
    data->to_index = (SIZE/2) - 1;
    pthread_create(&workers[0], 0, sorter, data);
    //establish the second sorting thread

    data = (parameters *) malloc (sizeof(parameters));
    data->from_index = (SIZE/2);
    data->to_index = SIZE - 1;
    pthread_create(&workers[1], 0, sorter, data);
    // now wait for the 2 sorting threads to finish

    for (i = 0; i < NUMBER_OF_THREADS - 1; i++)
    pthread_join(workers[i], NULL);
    //establish the merge thread

    data = (parameters *) malloc(sizeof(parameters));
    data->from_index = 0;
    data->to_index = (SIZE/2);
    pthread_create(&workers[2], 0, merger, data);
    // wait for the merge thread to finish

    pthread_join(workers[2], NULL);
    // output the sorted array


    return 0;
}
```

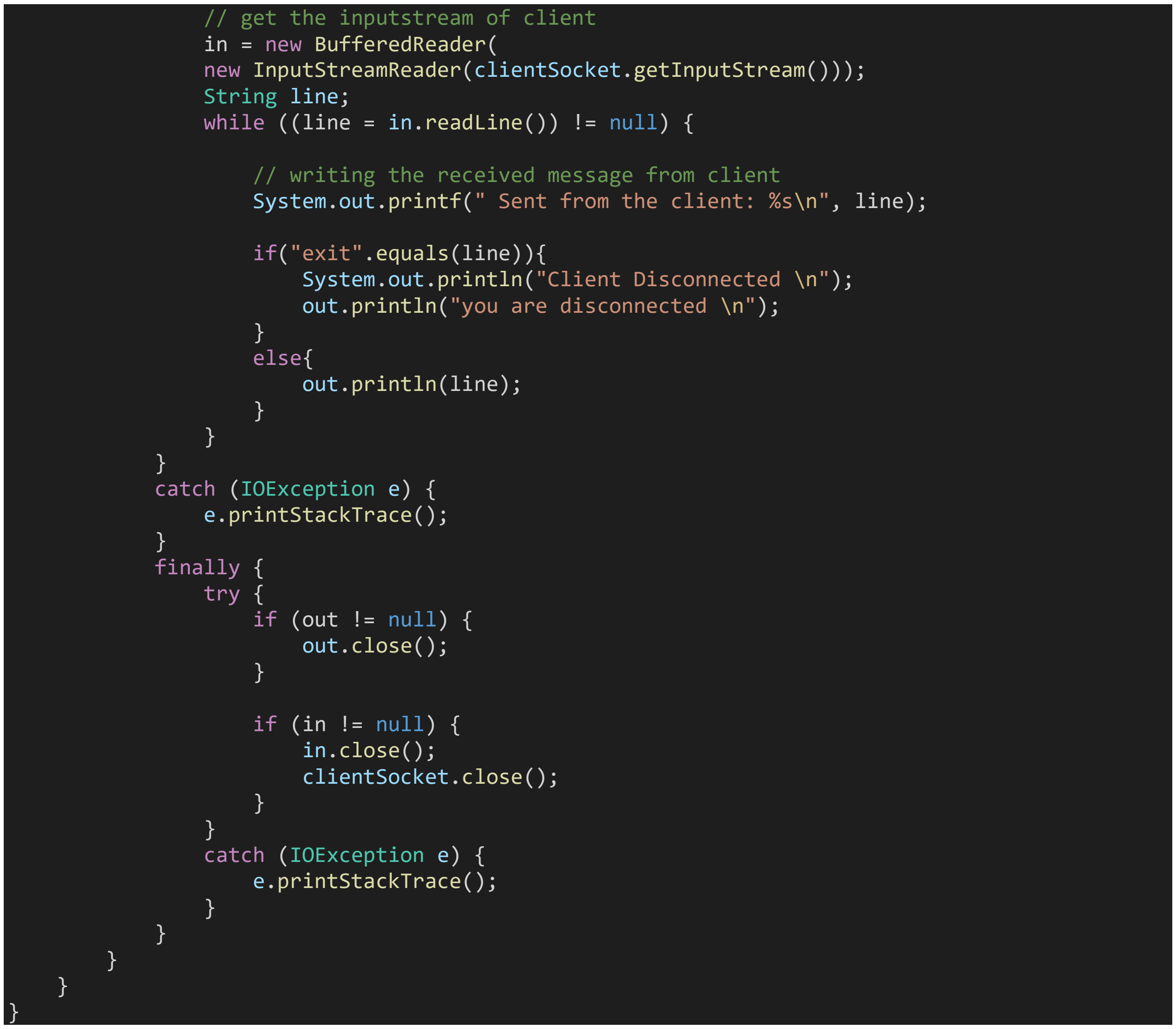
Ans to the question – D:

```c
#include<stdio.h>
#include<string.h>
#include<pthread.h>
#include<stdlib.h>
#include<unistd.h>
#include <semaphore.h>

sem_t x, y, z , rsem, wsem;
int readcount, writecount;

void initialize() {
    sem_init(&rsem, 0, 1);
    sem_init(&wsem, 0, 1);
    sem_init(&x, 0, 1);
    sem_init(&y, 0, 1);
    sem_init(&z, 0, 1);
    readcount = 0;
    writecount = 0;
}

void* reader(void* arg) {
    sem_wait(&z);
    sem_wait(&rsem);
    sem_wait(&x);

    printf("Reader is trying to enter\n");
    sleep(1);
    readcount++;

    if (readcount == 1) {
        sem_wait(&wsem);
    }

    sem_post(&x);
    sem_post(&rsem);
    sem_post(&z);

    printf("%d no Reader is inside \n", readcount);
    sleep(1);

    printf("Reader is leaving\n");
    sem_wait(&x);
    readcount--;
    if (readcount == 0) {
        sem_post(&wsem);
    }
    sem_post(&x);
}


void* writer(void* arg) {
    printf("Writer is trying to enter\n");
    sleep(1);

    sem_wait(&y);
    writecount++;

    if (writecount == 1) {
        sem_wait(&rsem);
    }

    sem_post(&y);
    sem_wait(&wsem);

    printf("%d no writer has entered the critical section\n", writecount);
    sleep(1);

    printf("writer is leaving\n");

    sem_post(&wsem);
    sem_wait(&y);
    writecount--;

    if (writecount == 0) {
        sem_post(&rsem);
    }
    sem_post(&y);
}


int main(){
    int r = 5;
    int w = 3;
    pthread_t rtid[r];
    pthread_t wtid[w];
    initialize();

    for (int i = 0; i < r; ++i){
        pthread_create(&(rtid[i]), NULL, &reader, NULL);
    }

    for (int i = 0; i < w; ++i){
        pthread_create(&(wtid[i]), NULL, &writer, NULL);
    }

    for (int i = 0; i < r; ++i){
        pthread_join(rtid[i], NULL);
    }

    for (int i = 0; i < w; ++i){
        pthread_join(wtid[i], NULL);
    }

    return 0;
}
```

Ans to the question – E:

```
//Client.java

import java.io.*;
import java.net.*;
import java.util.*;

class Client {
    // driver code
    public static void main(String[] args){
        // establish a connection by providing host and port
        // number
        try (Socket socket = new Socket("localhost", 1234)) {

            // writing to server
            PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

            // reading from server
            BufferedReader in
            = new BufferedReader(new InputStreamReader(socket.getInputStream()));

            // object of scanner class
            Scanner sc = new Scanner(System.in);
            String line = null;

            while (!"exit".equalsIgnoreCase(line)) {// reading from user
                line = sc.nextLine();

                // sending the user input to server
                out.println(line);
                out.flush();

                // displaying server reply
                System.out.println("Server replied "+ in.readLine());
            }

            // closing the scanner object
            sc.close();
        }
        catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

```java
//Server.java

import java.io.*;
import java.net.*;

class Server {
    public static void main(String[] args){
        ServerSocket server = null;try {
            // server is listening on port 1234
            server = new ServerSocket(1234);

            //server.setReuseAddress(true);
            // running infinite loop for getting
            // client request
            while (true) {
                // socket object to receive incoming client requests
                Socket c = server.accept();

                // Displaying that new client is connected to server
                System.out.println("New client connected
"+c.getInetAddress().getHostAddress());

                // create a new thread object
                ClientHandler clientSock = new ClientHandler(c);

                // This thread will handle the client separately
                new Thread(clientSock).start();
            }
        }
        catch (IOException e) {
            e.printStackTrace();
        }
        finally {
            if (server != null) {
                try {
                    server.close();
                }
                catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
    }

    // ClientHandler class
    private static class ClientHandler implements Runnable {
        private final Socket clientSocket;

        // Constructor
        public ClientHandler(Socket socket){
            this.clientSocket = socket;
        }

        public void run(){
            PrintWriter out = null;
            BufferedReader in = null;
            try {
                // get the outputstream of client
                out = new PrintWriter(
                clientSocket.getOutputStream(), true);
```



```java
                // get the inputstream of client
                in = new BufferedReader(
                new InputStreamReader(clientSocket.getInputStream()));
                String line;
                while ((line = in.readLine()) != null) {

                    // writing the received message from client
                    System.out.printf(" Sent from the client: %s\n", line);

                    if("exit".equals(line)){
                        System.out.println("Client Disconnected \n");
                        out.println("you are disconnected \n");
                    }
                    else{
                        out.println(line);
                    }
                }
            }
            catch (IOException e) {
                e.printStackTrace();
            }
            finally {
                try {
                    if (out != null) {
                        out.close();
                    }

                    if (in != null) {
                        in.close();
                        clientSocket.close();
                    }
                }
                catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
    }
}
```

Ans to the question – F:

```
//buffer.h

typedef int buffer_item;
#define BUFFER_SIZE 5
```

```
#include <stdlib.h>
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>
#include "buffer.h"

pthread_mutex_t mutex;
sem_t full, empty;
buffer_item buffer[BUFFER_SIZE];

int counter;
pthread_t tid;
pthread_attr_t attr;

void *producer(void *param);
void *consumer(void *param);

int insert_item(buffer_item);
int remove_item(buffer_item*) ;

void initializeData() {
    pthread_mutex_init(&mutex, NULL);
    sem_init(&full, 0, 0);
    sem_init(&empty, 0, BUFFER_SIZE);
    pthread_attr_init(&attr);
    counter = 0;
}

void *producer(void *param) {
    buffer_item item;
    while (1) {
        int rNum = rand() / 100000000;
        sleep(rNum);
        item = rand()%100;
        sem_wait(&empty);
        pthread_mutex_lock(&mutex);

        if (insert_item(item)) {
            fprintf(stderr, " Producer report error condition\n");
        }

        else {
            printf("producer produced: %d\n", item);
        }

        pthread_mutex_unlock(&mutex);
        sem_post(&full);
    }
```

```c
}

void *consumer(void *param) {
    buffer_item item;
    while (1) {
        int rNum = rand() / 1000000000;

        sleep(rNum);
        sem_wait(&full);
        pthread_mutex_lock(&mutex);

        if (remove_item(&item)) {
            fprintf(stderr, "Consumer report error condition\n");
        }
        else {
            printf("consumer consumed: %d\n", item);
        }
        pthread_mutex_unlock(&mutex);
        sem_post(&empty);
    }
}

int insert_item(buffer_item item) {
    if (counter < BUFFER_SIZE) {
        buffer[counter] = item;
        counter++;

        return 0;
    }
    else {
        return -1;
    }
}

int remove_item(buffer_item *item) {
    if (counter > 0) {
        *item = buffer[(counter - 1)];
        counter--;
        return 0;
    }
    else {
        return -1;
    }
}


int main(int argc, char *argv[]) {
    int i;

    if(argc != 4) {
        fprintf(stderr, "USAGE:./F <INT> <INT> <INT>\n");
        printf("Exiting the program\n");
        exit(0);
    }

    int sleeptime = atoi(argv[1]);
    int numProd = atoi(argv[2]);
    int numCons = atoi(argv[3]);

    initializeData();
    for (i = 0; i < numProd; i++) {
        pthread_create(&tid, &attr, producer, NULL);
    }

    for (i = 0; i < numCons; i++) {
        pthread_create(&tid, &attr, consumer, NULL);
    }

    sleep(sleeptime);

    printf("Exiting the program\n");
    exit(0);

    return 0;
}
```