OS Assignment 2

21K-3153

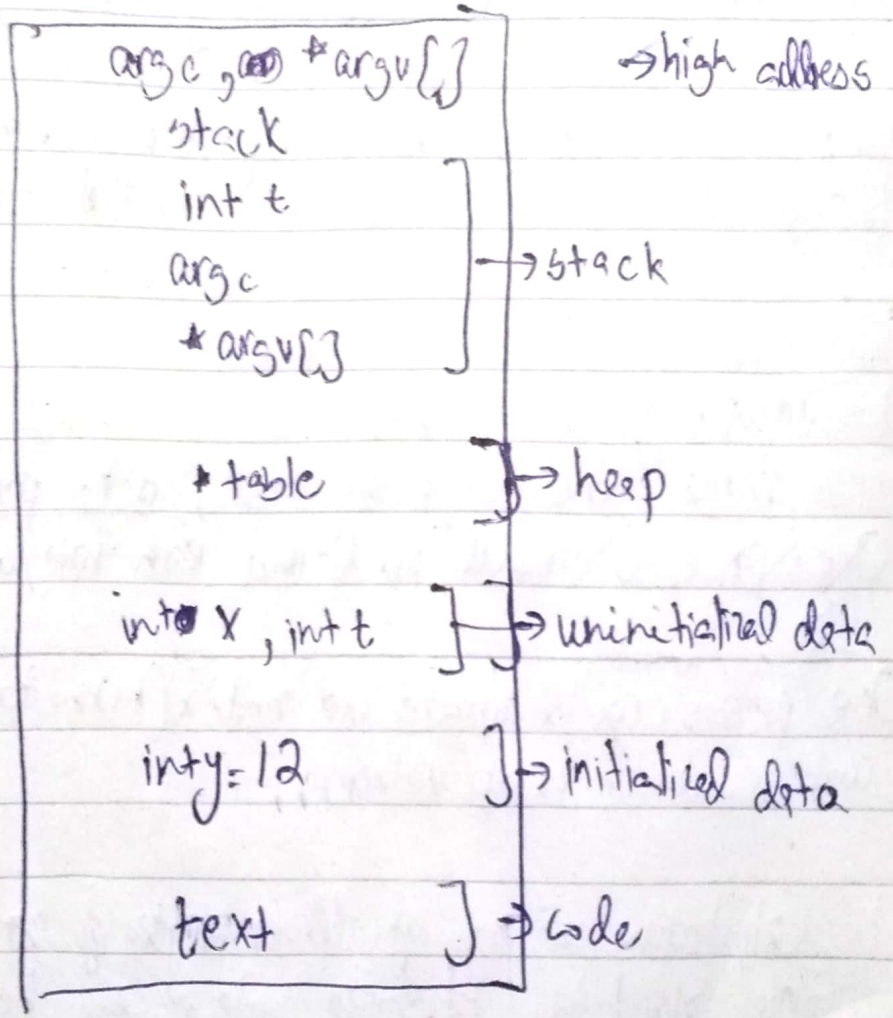
(Q1)

```
int main() {
    int id;
    int i = fork();
    if (i<0) {
        std:: cout << "Unsuccessful Child Process
                       Creation";
        exit(0);
    }
    else if (i==0) {
        id = getpid();
        std:: cout << "Parent id is " << getppid();
        for(int k=1; k<=10; k++) {
            if (k%2==0){
                continue;
            }
            std:: cout << k << endl; }
            std:: cout << "Child ends";
    }
else {
    waitpid (id, NULL, 0)
    for(int j=1; j<=10; j++) {
        if (j%2 != 0){
            continue;
        }
```

21K-3153

```
        std::cout << j;
    }
std::cout << " Parent ends";
}
```

(b)



21K-3153

(b)

While concurrent processing has benefits, it also has drawbacks

1) It can take longer for tasks to complete with higher tasks as each task needs it turn, putting other tasks on hold

2) Switching between processes is an overhead which takes more time and decrease efficiency and throughput

3) If synchronization isn't done properly, two or more process can change shared variable outside a critical section (data race)

(Q3)

(1) Transposing a matrix:

Since this is only one task, data parallelism will be used

One ~~process could pluck out each row/column and~~ thread could rotate the matrix (m×n to n×m) and another could swap values

(2) Calculating salary of all employees of company:

Since calculating ~~salary~~ salary can be considered one task, this can be done through data parallelism

One ~~process~~ thread could calculate salary for 1 to N/2 employees, while second ~~process~~ thread could do for N/2 to N employees.

21K-3153

(b)

While concurrent processing has benefits, it also has drawbacks

1) It can take longer for tasks to complete with higher tasks as each task needs it turn, putting other tasks on hold

2) Switching between process is an overhead which takes more time and decrease efficiency and throughput

3) If synchronization isn't done properly two or more process can change shared variable outside a critical section (data race)

(Q3)

(1) Transposing a matrix:

Since this is only one task, data parallelism will be used. One thread could rotate the matrix (mxn to nxm) and another could swap values

(2) Calculating salary of all employees of company:

Since calculating salary can be considered one task, this can be done through data parallelism. One thread could calculate salary for 1 to N/2 employees, while second thread could do for N/2 to N employees.

2) Taking input from IOT camera for a single detection

This can be done through both data and task parallelism
One thread could store data from 1 to N/2 cameras, one could from N/2 to N cameras. Then other threads could perform various algorithms on the data sets according to environments to detect abnormality.

3) Taking input from IOT cameras for a single abnormal detection
Data parallelism.
One thread could store data and apply detection algorithm for one single abnormal activity from 1 to N/2 cameras.
A second threads could do the same for N/2 - N or more cameras

4) Taking input in library for multiple modules
Task parallelism
As there will be different detection algorithms/ tasks to be performed on each camera recording, different threads will store camera recording and apply detection algorithms separately

(V) National ID card making procedure
Both task and data parallelism

Data parallelism to split number of people (1 to N/2, N/2 ... N). Task DP

Task parallelism happening inside data parallelism for different counters (photo booth, fingerprint record, signs etc)

~~#include <pthread.h>~~

void registration (string name)
~~void announcements (string ann~~
~~void sponsors (string sponsors)~~
~~void~~

```
#include <pthread.h>
#include <stdio.h>
void registration (char * name)
char * announcements ();
char * sponsors ();
char * query (char * question)

int main(){
    pthread_t procom[4];
    pthread_attr_t attr;

    pthread_attr_init(&attr);    //default attributes
    int i; char n[100]; char q[100];
    for (i=0; i<100; i++){
        printf("enter name");
        scanf("%s", &n);
        pthread_create(&thread[0], &a
//volunteer 1    pthread_create(&procom[0], &attr, registration, n)
        printf("if query, enter query, if not, enter n);
        scanf("%s", &q);
//volunteer 4    pthread_create(&procom[3], &attr, query, q)
    }
//volunteer 2  pthread_create(&procom[1], &attr, announcement, NULL)
//volunteer 3  pthread_create(&procom[2], &attr, sponsors, NULL)

    for(i=0; i<4; i++)
        pthread_join(procom[i], NULL);
}
```

registration

```
void announcements(char *name)
{
    printf("%s, you have been registered", name);
    printf("Welcome to PROcom");

    pthread_exit(0);

}

void announcements()
    printf("Listing all important announcements:
    1) Please follow rules
    2) No destiny
    3) Take ID card from reception");

    pthread_exit(0);
}

void sponsors(){
    printf("Here are your sponsors of PROcom":

                .
                .
                .
                .

            Thanks to all sponsors");
    pthread_exit(0);
}
```

```
char * query(char *q){

    if (strcmp(q, "n") == 0) {
        return "Enjoy Procon";
    }
    else {
        for (int i = 0; i < queryDatabaseLength; i++) {
            if (strcmp(q, queryDB[i]) == 0) {
                return "answer is %s", answerDB[i];
            }
            else
            {
            }
        }
        pthread_exit(0);
}
```