Cheatsheet Name: Ms Rodsy Tahmid id: 20101021

Lab_Asg01 2d)

```
#include <stdio.h>
#include <string.h>
int isUpdated(char email[]) {
   char *domain = strstr(email, "@");
   if (domain != NULL) {
      if (strstr(domain, "kaaj.com") != NULL) {
         return 0;
      }
   }
   return 1;
}
int main() {
   char email[100];
   printf("Enter email address: ");
   scanf("%s", email);
   if (isUpdated(email)) {
      printf("Email address is okay\n");
   } else {
      printf("Email address is outdated\n");
   }
   return 0;
}
// // // // // // // // // // // // // // // // // // // // // // // // //
//
```

Lab_Asg01 1a)

```
-> touch 20101021_1.txt
-> mkdir Rodsy1
-> mv 20101021_1.txt Rodsy1
-> cp Rodsy1/20101021_1.txt Rodsy2
-> cd Rodsy3
-> ls -l
-> chmod go=rx *
-> cd ..
-> ls -R
```

Lab_Asg02 sys call oddveven task04)

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
int main(int argc, char *argv[]){
        int arr[argc-1], odd[argc-1], even[argc-1];
        int a;
        int k=0,o=0,e=0,count1=0,count2=0;
        for (int i=1; i<argc; i++){
                arr[k] = atoi(argv[i]);
                k++;
        }
        for (int j=0; j<argc-1 ; j++){
                if (arr[j]%2!=0){
                        odd[o] = arr[j];
                        o++;
                        count1++;
                }
                else if (arr[j]%2==0){
                        even[e] = arr[j];
                        e++;
                        count2++;
                }
        }
        printf("The odd numbers are:\n");
        for (int l=0; l<count1; l++){
                printf("%d\n",odd[l]);
        }
        printf("The even numbers are:\n");
        for (int p=0; p<count2; p++){
                printf("%d\n",even[p]);
        }
}
```

Lab_Asg02 thread task01)

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>
#include <string.h>
int count=0;
void *thread_function(void *thread_id){
        count++;
        printf("thread-%d running\n",count);
        int thrd_id;
        for (int j=0; j<5; j++){
                sleep(0.002);
        }
        printf("thread-%d closed\n",count);
}
int main(){
        pthread_t thread_arr[5];
        for (int i=0; i<5; i++){
        pthread_create(&thread_arr[i],NULL,thread_function
,NULL);
                pthread_join(thread_arr[i],NULL);
```

```
void *ShopOwner(void *sho){
  // Shopowner mane task01 er consumer
   for(int i = 0; i < MaxCrops; i++){
      sem_wait(&full); //wait korbo warehouse (buffer) jodi
empty thake
      pthread_mutex_lock(&mutex); // mutex ta lock korte

       // buffer e crops (item) remove korte
      char item = warehouse[out];
      printf("Shop Owner %d: Remove crops %c from %d\n",
*((int *)sho), item, out);
      warehouse[out] = 'N'; // slot ta N e reset korte

      // out index update aar dorkare wrap around korte
      out = (out + 1) % warehouseSize;

      pthread_mutex_unlock(&mutex); // mutex ta unlock korte
      sem_post(&empty); // warehouse (buffer) je aar full na
sheta signal korte
   }

   printf("ShopOwner%d:", *((int *)sho));
```

```
        }
}
// // // // // // // // // // // // // // // // // // // // // // // // // //
// // //
```

Lab_Asg04 task02)

```
#include <pthread.h>
#include <semaphore.h>
#include <stdio.h>
#include <stdlib.h> // extra add korsi
#include <time.h> // extra add korsi
#define MaxCrops 5 // Maximum crops a Farmer can produce or a Shpoowner can take
#define warehouseSize 5 // Size of the warehouse
sem_t empty;
sem_t full;
int in = 0;
int out = 0;
char crops[warehouseSize]={'R','W','P','S','M'}; //indicating room for different crops
char warehouse[warehouseSize]={'N','N','N','N','N'}; //initially all the room is empty
pthread_mutex_t mutex;
void *Farmer(void *far)
{
   // farmer mane task01 er producer
   for(int i = 0; i < MaxCrops; i++){
      sem_wait(&empty); //wait korbo warehouse (buffer) jodi full thake
      pthread_mutex_lock(&mutex); // mutex ta lock korte

      // buffer e crops (item) insert korte
      warehouse[in]=crops[in];
      printf("Farmer %d: Insert crops %c at %d\n", *((int *)far), warehouse[in], in);

      // insert index update aar dorkare wrap around korte
      in = (in + 1) % warehouseSize;

      pthread_mutex_unlock(&mutex); // mutex ta unlock korte
      sem_post(&full); // warehouse (buffer) je aar empty na sheta signal korte
   }

   printf("Farmer%d:", *((int *)far));
   for(int g=0; g<warehouseSize; g++){
      printf(" %c", warehouse[g]);
   }
   printf("\n");
}
   for(int w=0; w<warehouseSize; w++){
      printf(" %c", warehouse[w]);
   }

   printf("\n");
}

int main()
{
   /*initializing thread,mutex,semaphore
   */
   pthread_t Far[5], Sho[5];
   pthread_mutex_init(&mutex, NULL);
   sem_init(&empty, 0, warehouseSize); //when the warehouse is full thread will wait
   sem_init(&full, 0, 0); //when the warehouse is empty thread will wait
   int a[5] = {1, 2, 3, 4, 5}; //Just used for numbering the Farmer and ShopOwner

  // Farmer (producer) thread create korte
   for(int i=0; i<5; i++){
      pthread_create(&Far[i], NULL, (void *)Farmer, (void *)&a[i]);
   }

   // ShopOwner (consumer) thread create korte
   for(int i=0; i<5; i++){
      pthread_create(&Sho[i], NULL, (void *)ShopOwner, (void *)&a[i]);
   }
   // shob Farmer thread jate shesh hoi shejonno wait kora hochche
   for(int i=0; i<5; i++){
      pthread_join(Far[i], NULL);
   }

   // shob ShopOwnner(producer) thread jate shesh hoi shejonno wait kora hochche
   for(int i=0; i<5; i++){
      pthread_join(Sho[i], NULL);
   }

   //  Closing or destroying mutex and semaphore
   pthread_mutex_destroy(&mutex);
   sem_destroy(&empty);
   sem_destroy(&full);

   return 0;
}
```

- ➔ gcc -o task01 task01.c -pthread
- ➔ ./task01