NTU

INNOVATE AND LEAD

National Textile University Department of Computer Science

Subject: Operating System

Submitted to: Sir Nasir Mahmood

Submitted by: Nimra Tanveer

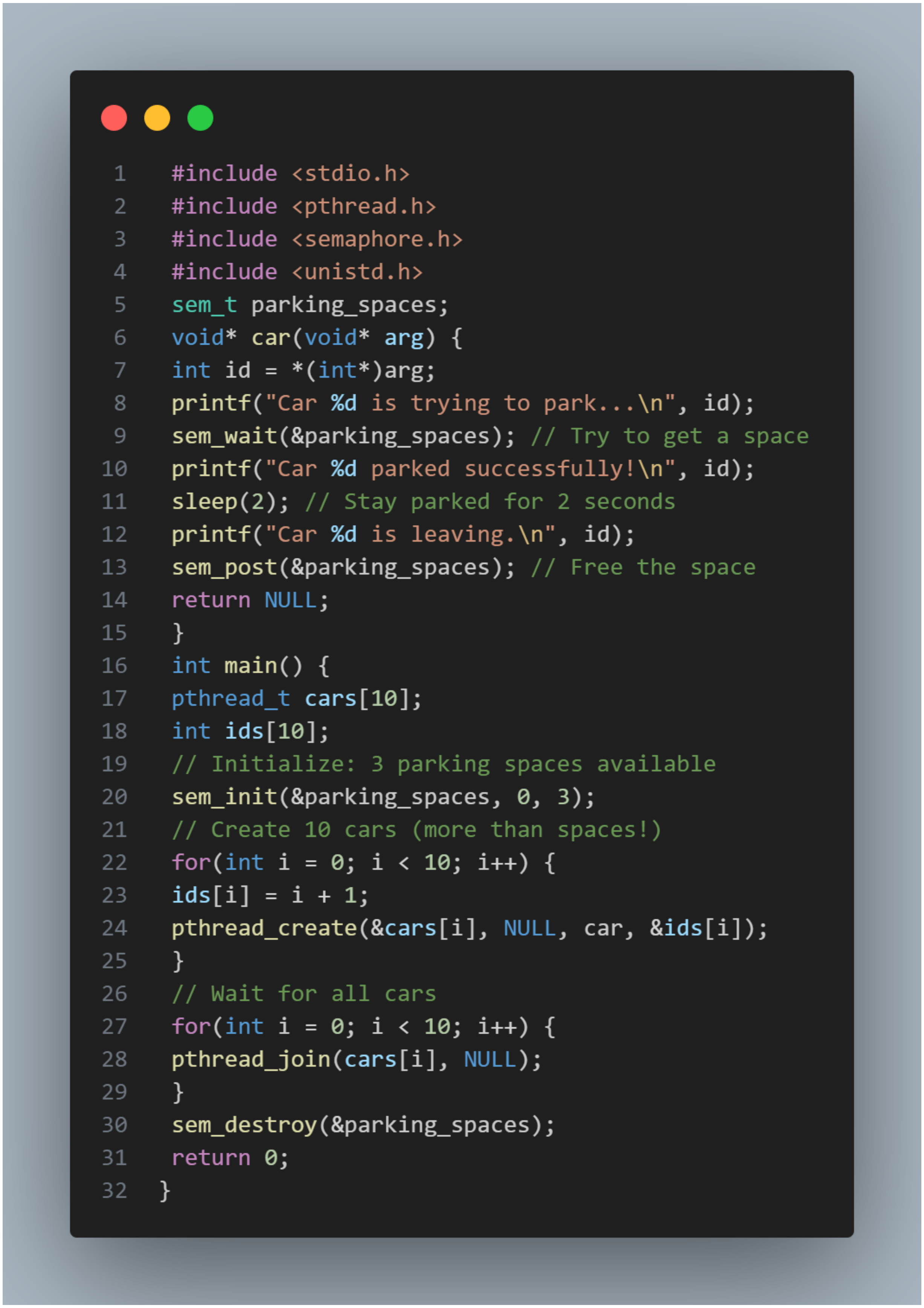
Reg number:23-NTU-CS-1201

Section: BS SE 5th (A)

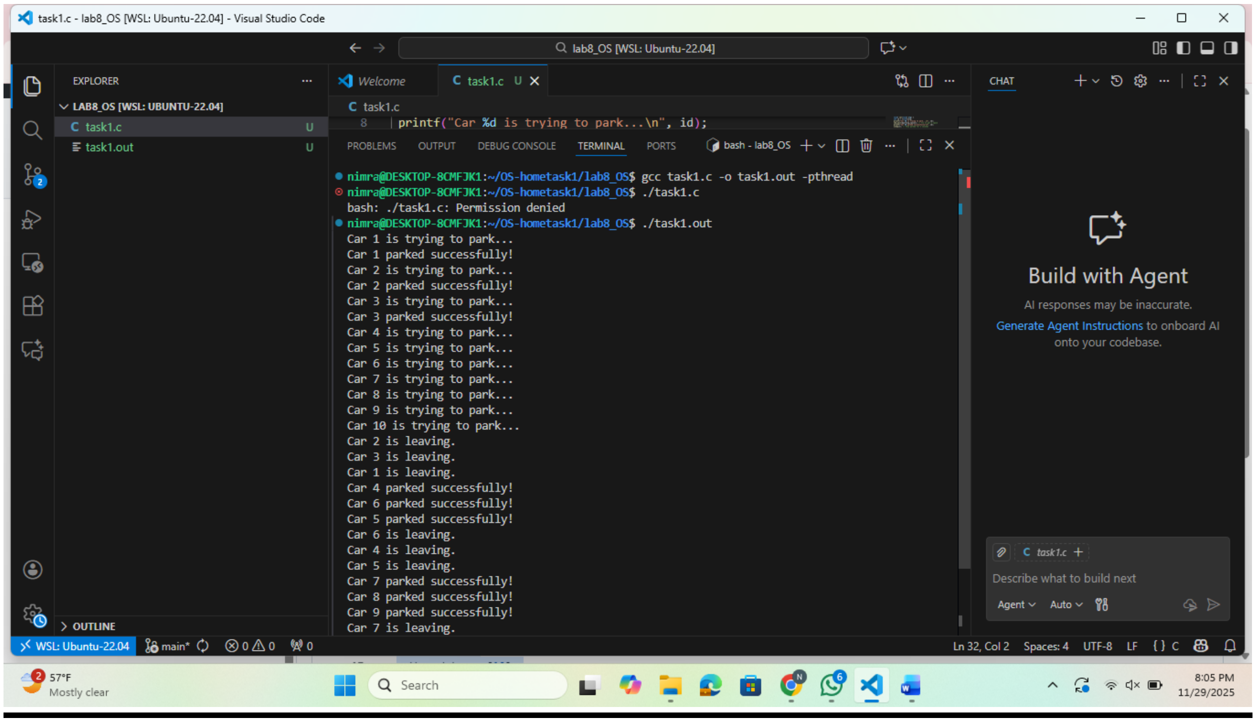
LAB MANNUAL 10

Task 1:

- **This program controls parking using a semaphore with 3 spaces.**
- **Ten cars (threads) try to park, but only 3 can park at once while others wait.**



```c
 #include <stdio.h>
 #include <pthread.h>
 #include <semaphore.h>
 #include <unistd.h>
 sem_t parking_spaces;
 void* car(void* arg) {
 int id = *(int*)arg;
 printf("Car %d is trying to park...\n", id);
 sem_wait(&parking_spaces); // Try to get a space
 printf("Car %d parked successfully!\n", id);
 sleep(2); // Stay parked for 2 seconds
 printf("Car %d is leaving.\n", id);
 sem_post(&parking_spaces); // Free the space
 return NULL;
 }
 int main() {
 pthread_t cars[10];
 int ids[10];
 // Initialize: 3 parking spaces available
 sem_init(&parking_spaces, 0, 3);
 // Create 10 cars (more than spaces!)
 for(int i = 0; i < 10; i++) {
 ids[i] = i + 1;
 pthread_create(&cars[i], NULL, car, &ids[i]);
 }
 // Wait for all cars
 for(int i = 0; i < 10; i++) {
 pthread_join(cars[i], NULL);
 }
 sem_destroy(&parking_spaces);
 return 0;
}
```



task1.c - lab8_OS [WSL: Ubuntu-22.04] - Visual Studio Code
lab8_OS [WSL: Ubuntu-22.04]
EXPLORER
LAB8_OS [WSL: UBUNTU-22.04]
task1.c U
task1.out U
OUTLINE
Welcome
task1.c U
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
bash - lab8_OS
CHAT
Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
Describe what to build next
Agent
Auto
WSL: Ubuntu-22.04
main*
Ln 32, Col 2
Spaces: 4
UTF-8
LF
{ } C
57°F
Mostly clear
Search
8:05 PM
11/29/2025

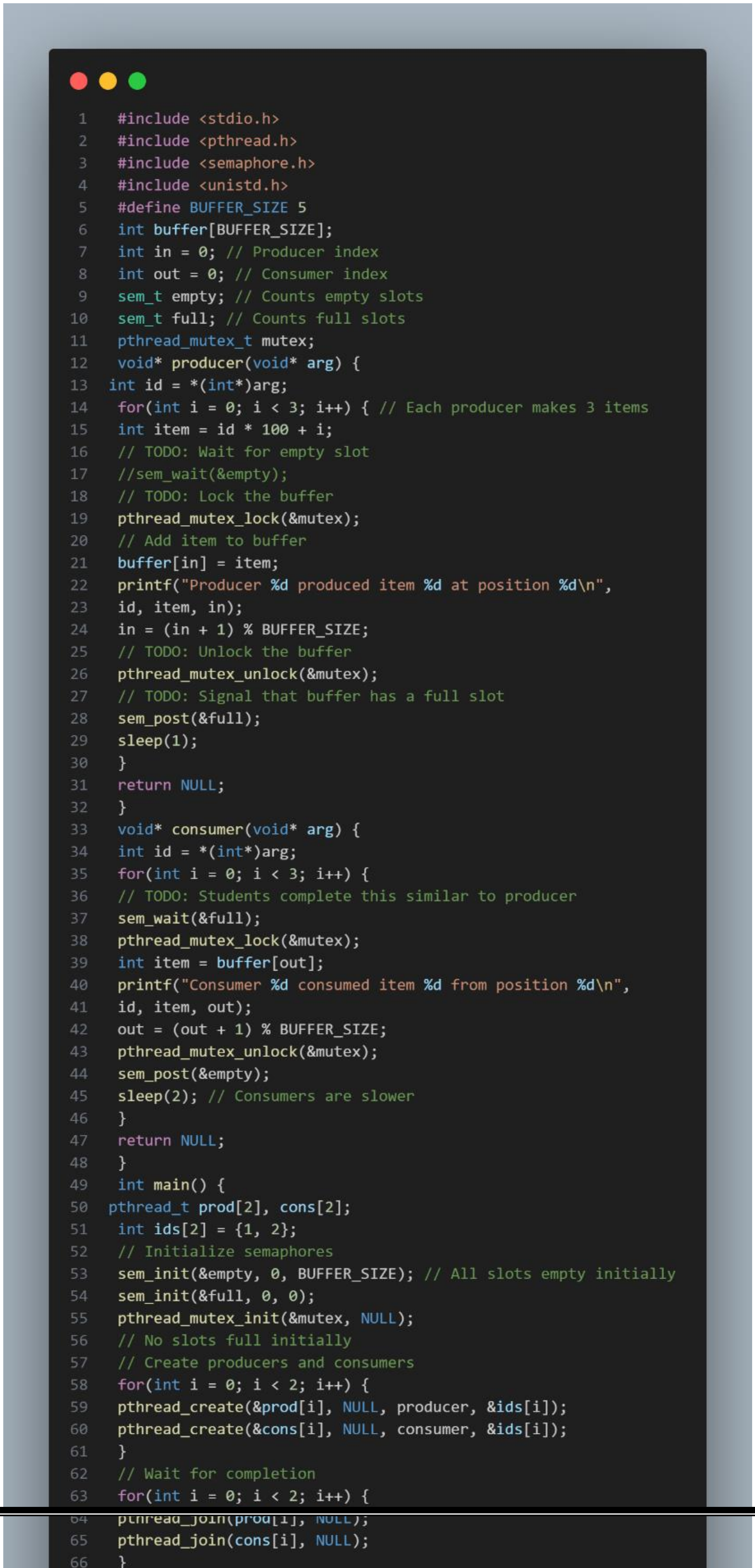
```
8   printf("Car %d is trying to park...\n", id);
```

```
nimra@DESKTOP-8CMFJK1:~/OS-hometask1/lab8_OS$ gcc task1.c -o task1.out -pthread
nimra@DESKTOP-8CMFJK1:~/OS-hometask1/lab8_OS$ ./task1.c
bash: ./task1.c: Permission denied
nimra@DESKTOP-8CMFJK1:~/OS-hometask1/lab8_OS$ ./task1.out
Car 1 is trying to park...
Car 1 parked successfully!
Car 2 is trying to park...
Car 2 parked successfully!
Car 3 is trying to park...
Car 3 parked successfully!
Car 4 is trying to park...
Car 5 is trying to park...
Car 6 is trying to park...
Car 7 is trying to park...
Car 8 is trying to park...
Car 9 is trying to park...
Car 10 is trying to park...
Car 2 is leaving.
Car 3 is leaving.
Car 1 is leaving.
Car 4 parked successfully!
Car 6 parked successfully!
Car 5 parked successfully!
Car 6 is leaving.
Car 4 is leaving.
Car 5 is leaving.
Car 7 parked successfully!
Car 8 parked successfully!
Car 9 parked successfully!
Car 7 is leaving.
```

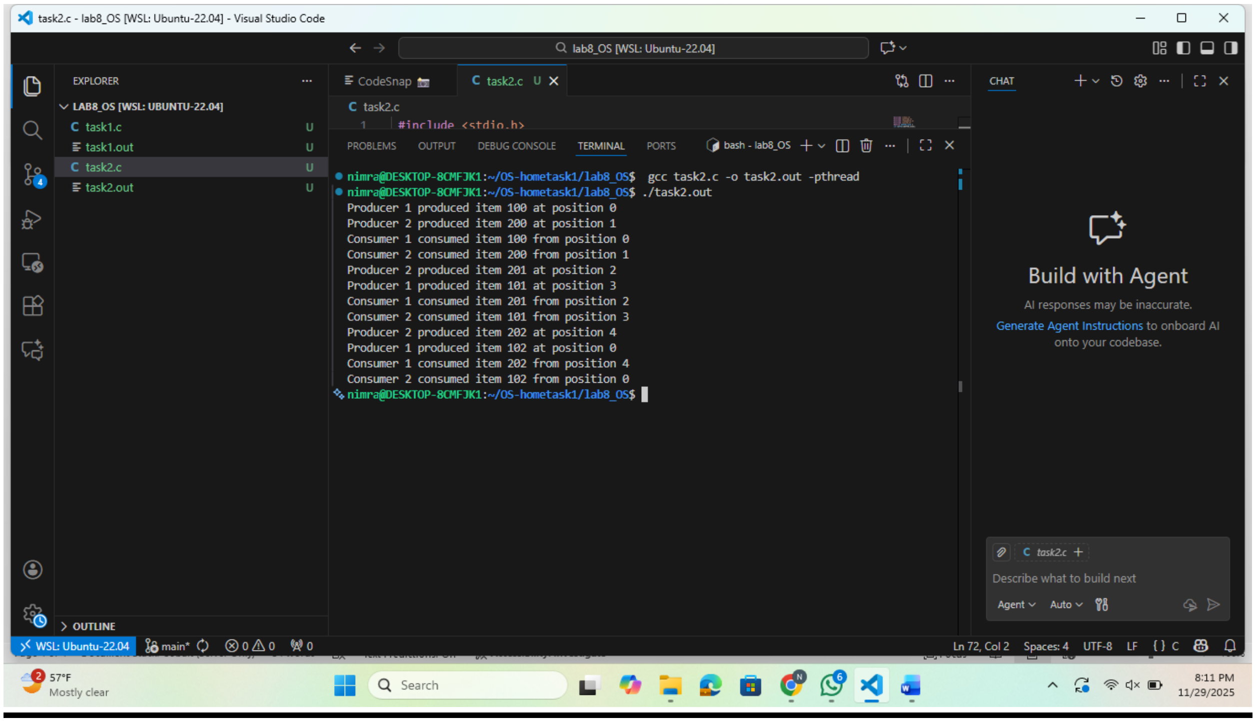
Task 2:

- **This program shows how producers add items to a shared buffer and consumers remove them using semaphores and a mutex.**

- **Semaphores control empty or full spaces, while the mutex prevents two threads from accessing the buffer at the same time.**



```c
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>
#define BUFFER_SIZE 5
int buffer[BUFFER_SIZE];
int in = 0; // Producer index
int out = 0; // Consumer index
sem_t empty; // Counts empty slots
sem_t full; // Counts full slots
pthread_mutex_t mutex;
void* producer(void* arg) {
int id = *(int*)arg;
for(int i = 0; i < 3; i++) { // Each producer makes 3 items
int item = id * 100 + i;
// TODO: Wait for empty slot
//sem_wait(&empty);
// TODO: Lock the buffer
pthread_mutex_lock(&mutex);
// Add item to buffer
buffer[in] = item;
printf("Producer %d produced item %d at position %d\n",
id, item, in);
in = (in + 1) % BUFFER_SIZE;
// TODO: Unlock the buffer
pthread_mutex_unlock(&mutex);
// TODO: Signal that buffer has a full slot
sem_post(&full);
sleep(1);
}
return NULL;
}
void* consumer(void* arg) {
int id = *(int*)arg;
for(int i = 0; i < 3; i++) {
// TODO: Students complete this similar to producer
sem_wait(&full);
pthread_mutex_lock(&mutex);
int item = buffer[out];
printf("Consumer %d consumed item %d from position %d\n",
id, item, out);
out = (out + 1) % BUFFER_SIZE;
pthread_mutex_unlock(&mutex);
sem_post(&empty);
sleep(2); // Consumers are slower
}
return NULL;
}
int main() {
pthread_t prod[2], cons[2];
int ids[2] = {1, 2};
// Initialize semaphores
sem_init(&empty, 0, BUFFER_SIZE); // All slots empty initially
sem_init(&full, 0, 0);
pthread_mutex_init(&mutex, NULL);
// No slots full initially
// Create producers and consumers
for(int i = 0; i < 2; i++) {
pthread_create(&prod[i], NULL, producer, &ids[i]);
pthread_create(&cons[i], NULL, consumer, &ids[i]);
}
// Wait for completion
for(int i = 0; i < 2; i++) {
pthread_join(prod[i], NULL);
pthread_join(cons[i], NULL);
}
```



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

```
nimra@DESKTOP-8CMFJK1:~/OS-hometask1/lab8_OS$  gcc task2.c -o task2.out -pthread
nimra@DESKTOP-8CMFJK1:~/OS-hometask1/lab8_OS$ ./task2.out
Producer 1 produced item 100 at position 0
Producer 2 produced item 200 at position 1
Consumer 1 consumed item 100 from position 0
Consumer 2 consumed item 200 from position 1
Producer 2 produced item 201 at position 2
Producer 1 produced item 101 at position 3
Consumer 1 consumed item 201 from position 2
Consumer 2 consumed item 101 from position 3
Producer 2 produced item 202 at position 4
Producer 1 produced item 102 at position 0
Consumer 1 consumed item 202 from position 4
Consumer 2 consumed item 102 from position 0
nimra@DESKTOP-8CMFJK1:~/OS-hometask1/lab8_OS$ 
```

Build with Agent

AI responses may be inaccurate.

Generate Agent Instructions to onboard AI onto your codebase.

task2.c

Describe what to build next

Agent Auto