perplexity

not the core, complete code with libraries

Here are **complete, minimal C codes** for your OS lab programs, each with required headers, main function, and basic input/output. These are ready to copy, compile, and run. Each program is kept as short as possible while remaining fully functional.

4. CPU Scheduling Algorithms

a) FCFS

```
#include <stdio.h>
int main() {
    int n, bt[^10], wt[^10], tat[^10], i;
    float awt=0, atat=0;
    printf("Enter number of processes: ");
    scanf("%d",&n);
    printf("Enter burst times:\n");
    for(i=0;i<n;i++) scanf("%d",&bt[i]);
    wt[^0]=0;
    for(i=1;i<n;i++) wt[i]=wt[i-1]+bt[i-1];
    for(i=0;i<n;i++) {
        tat[i]=wt[i]+bt[i];
        awt+=wt[i]; atat+=tat[i];
    }
    printf("Avg WT=%.2f\nAvg TAT=%.2f\n",awt/n,atat/n);
    return 0;
}
```

b) SJF (Non-preemptive)

```
#include <stdio.h>
int main() {
    int n, bt[^10], wt[^10], tat[^10], i, j, temp;
    float awt=0, atat=0;
    printf("Enter number of processes: ");
    scanf("%d",&n);
    printf("Enter burst times:\n");
    for(i=0;i<n;i++) scanf("%d",&bt[i]);
    for(i=0;i<n-1;i++)
        for(j=i+1;j<n;j++)
            if(bt[i]>bt[j]) { temp=bt[i]; bt[i]=bt[j]; bt[j]=temp; }
    wt[^0]=0;
    for(i=1;i<n;i++) wt[i]=wt[i-1]+bt[i-1];
    for(i=0;i<n;i++) {
        tat[i]=wt[i]+bt[i];
```

```
        awt+=wt[i]; atat+=tat[i];
    }
    printf("Avg WT=%.2f\nAvg TAT=%.2f\n",awt/n,atat/n);
    return 0;
}
```

c) Priority

```
#include <stdio.h>
int main() {
    int n, bt[^10], pri[^10], wt[^10], tat[^10], i, j, temp;
    float awt=0, atat=0;
    printf("Enter number of processes: ");
    scanf("%d",&n);
    printf("Enter burst time and priority:\n");
    for(i=0;i<n;i++) scanf("%d%d",&bt[i],&pri[i]);
    for(i=0;i<n-1;i++)
        for(j=i+1;j<n;j++)
            if(pri[i]>pri[j]) {
                temp=pri[i]; pri[i]=pri[j]; pri[j]=temp;
                temp=bt[i]; bt[i]=bt[j]; bt[j]=temp;
            }
    wt[^0]=0;
    for(i=1;i<n;i++) wt[i]=wt[i-1]+bt[i-1];
    for(i=0;i<n;i++) {
        tat[i]=wt[i]+bt[i];
        awt+=wt[i]; atat+=tat[i];
    }
    printf("Avg WT=%.2f\nAvg TAT=%.2f\n",awt/n,atat/n);
    return 0;
}
```

d) Round Robin

```
#include <stdio.h>
int main() {
    int n, bt[^10], rem[^10], tq, t=0, done;
    printf("Enter number of processes: ");
    scanf("%d",&n);
    printf("Enter burst times:\n");
    for(int i=0;i<n;i++) { scanf("%d",&bt[i]); rem[i]=bt[i]; }
    printf("Enter time quantum: ");
    scanf("%d",&tq);
    do {
        done=1;
        for(int i=0;i<n;i++) {
            if(rem[i]>0) {
                done=0;
                if(rem[i]>tq) { t+=tq; rem[i]-=tq; }
                else { t+=rem[i]; rem[i]=0; }
            }
        }
    } while(!done);
```

```
    printf("Total time: %d\n", t);
    return 0;
}
```

5. Semaphore Port Control

```
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
sem_t s;
void* port(void* a) {
    sem_wait(&s);
    printf("Port accessed\n");
    sem_post(&s);
    return NULL;
}
int main() {
    sem_init(&s,0,2);
    pthread_t t[^3];
    for(int i=0;i<3;i++) pthread_create(&t[i],0,port,0);
    for(int i=0;i<3;i++) pthread_join(t[i],0);
    return 0;
}
```

6. Concurrent Threads (pthreads)

```
#include <stdio.h>
#include <pthread.h>
void* run(void* a) {
    printf("Thread %d running\n", *(int*)a);
    return NULL;
}
int main() {
    pthread_t t[^2];
    int a=1, b=2;
    pthread_create(&t[^0],0,run,&a);
    pthread_create(&t[^1],0,run,&b);
    pthread_join(t[^0],0);
    pthread_join(t[^1],0);
    return 0;
}
```

7. Producer-Consumer (Semaphores)

```
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
sem_t e,f,m; int buf=0;
```

```
void* prod() { sem_wait(&e); sem_wait(&m); buf++; printf("Produced\n"); sem_post(&m); sem
void* cons() { sem_wait(&f); sem_wait(&m); buf--; printf("Consumed\n"); sem_post(&m); sem
int main() {
    sem_init(&e,0,1); sem_init(&f,0,0); sem_init(&m,0,1);
    pthread_t p,c;
    pthread_create(&p,0,prod,0); pthread_create(&c,0,cons,0);
    pthread_join(p,0); pthread_join(c,0);
    return 0;
}
```

8. Memory Allocation (First Fit)

```
#include <stdio.h>
int main() {
    int b[^3]={5,10,15},p[^2]={10,5},a[^2]={-1,-1},i,j;
    for(i=0;i<2;i++)
        for(j=0;j<3;j++)
            if(b[j]>=p[i]) { a[i]=j; b[j]-=p[i]; break; }
    for(i=0;i<2;i++) printf("P%d->B%d\n",i+1,a[i]+1);
    return 0;
}
```

9. Page Replacement (FIFO)

```
#include <stdio.h>
int main() {
    int f[^3]={-1,-1,-1},p[^5]={1,2,3,4,1},i,j,k=0,c=0;
    for(i=0;i<5;i++) {
        for(j=0;j<3;j++) if(f[j]==p[i]) break;
        if(j==3) { f[k++%3]=p[i]; c++; }
    }
    printf("Page faults: %d\n",c);
    return 0;
}
```

10. Banker's Algorithm (Simple)

```
#include <stdio.h>
int main() {
    int m[^2][^2]={{3,2},{2,2}},a[^2][^2]={{2,0},{1,2}},v[^2]={1,1},n[^2][^2],f[^2]={0,0}
    for(i=0;i<2;i++)for(j=0;j<2;j++)n[i][j]=m[i][j]-a[i][j];
    while(k<2)
        for(i=0;i<2;i++)
            if(!f[i]&&n[i][^0]<=v[^0]&&n[i][^1]<=v[^1]){v[^0]+=a[i][^0];v[^1]+=a[i][^1];f
    printf(k==2?"Safe\n":"Unsafe\n");
    return 0;
}
```

11. File Allocation (Sequential)

```
#include <stdio.h>
int main() {
    int s=0,i;
    for(i=0;i<3;i++) { printf("%d-%d\n",s,s+4); s+=5; }
    return 0;
}
```

12. Disk Scheduling (FCFS)

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int r[^3]={10,20,30},h=5,i,m=0;
    for(i=0;i<3;i++) { m+=abs(r[i]-h); h=r[i]; }
    printf("Total head movement: %d\n",m);
    return 0;
}
```

Tip:

- For pthread/sem programs, compile with `gcc code.c -lpthread`.
- You can expand these for more processes or features as needed.
- For more, see [OS lab C examples][1].

⁂

1. https://www.slideshare.net/slideshow/os-lab-file-c-programs/42055538