b) Earliest-deadline Firs

```
#include <stdio.h>

struct Process {
   int id;
   int burst_time;
   int period;
   int remaining_time;
   int next_deadline;
};

int main() {
   int n, hyper_period = 1, time = 0;
   struct Process p[10];

   printf("Enter number of real-time processes: ");
   scanf("%d", &n);

   for(int i = 0; i < n; i++) {
      printf("Enter burst time and period for P[%d]: ", i+1);
      scanf("%d %d", &p[i].burst_time, &p[i].period);
      p[i].id = i + 1;
      p[i].remaining_time = p[i].burst_time;
      p[i].next_deadline = p[i].period;
      hyper_period *= p[i].period; // for simplicity, assume all periods are relatively prime
   }

   printf("\nEDF Scheduling up to time %d:\n", hyper_period);
   printf("Time\tProcess\n");

   while(time < hyper_period) {
      for(int i = 0; i < n; i++) {
         if(time != 0 && time % p[i].period == 0) {
            p[i].remaining_time = p[i].burst_time;
            p[i].next_deadline = time + p[i].period;
         }
      }

      int earliest = -1;
      for(int i = 0; i < n; i++) {
         if(p[i].remaining_time > 0) {
            if(earliest == -1 || p[i].next_deadline < p[earliest].next_deadline) {
               earliest = i;
            }
         }
      }
```

```
        if(earliest != -1) {
            printf("%d\tP[%d]\n", time, p[earliest].id);
            p[earliest].remaining_time--;
        } else {
            printf("%d\tIdle\n", time);
        }

        time++;
    }

    return 0;
}
```

Output

```
Enter number of real-time processes: 1
Enter burst time and period for P[1]: 3
3

EDF Scheduling up to time 3:
Time    Process
0   P[1]
1   P[1]
2   P[1]


=== Code Execution Successful ===
```