✅ **EX.NO: 3A – FCFS Scheduling Algorithm**

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

struct Process {
   char name[5];
   int bt, at, ft, wt, tat;
};

void main() {
   int n, i, j, time = 0;
   float avg_wt = 0, avg_tat = 0;
   struct Process p[10], temp;

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

   printf("Enter process name and burst time: ");
   for (i = 0; i < n; i++)
      scanf("%s%d", p[i].name, &p[i].bt);

   printf("Enter arrival times: ");
   for (i = 0; i < n; i++)
      scanf("%d", &p[i].at);

   // Sort by arrival time
```

```
for (i = 0; i < n - 1; i++)
   for (j = i + 1; j < n; j++)
      if (p[i].at > p[j].at) {
         temp = p[i];
         p[i] = p[j];
         p[j] = temp;
      }

for (i = 0; i < n; i++) {
   if (time < p[i].at) time = p[i].at;
   p[i].wt = time - p[i].at;
   p[i].ft = time + p[i].bt;
   p[i].tat = p[i].ft - p[i].at;
   time = p[i].ft;
   avg_wt += p[i].wt;
   avg_tat += p[i].tat;
}

// Gantt Chart
printf("\nGANTT CHART\n|");
for (i = 0; i < n; i++) printf(" %s |", p[i].name);
printf("\n0");
for (i = 0; i < n; i++) printf("   %d", p[i].ft);

// Output
printf("\n\nFCFS Scheduling:\n");
```

```
  printf("Process\tBT\tAT\tWT\tFT\tTAT\n");
  for (i = 0; i < n; i++)
    printf("%s\t%d\t%d\t%d\t%d\t%d\n", p[i].name, p[i].bt, p[i].at, p[i].wt, p[i].ft, p[i].tat);

  printf("Avg WT: %.2f\tAvg TAT: %.2f\n", avg_wt / n, avg_tat / n);
}
```

✅ **EX.NO: 3B – SJF Scheduling Algorithm**

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

void main() {
  int n, bt[10], at[10], wt[10], tat[10], p[10], i, j, temp;
  float awt = 0, atat = 0;
  clrscr();

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

  for (i = 0; i < n; i++) {
    p[i] = i;
    printf("P%d BT: ", i);
    scanf("%d", &bt[i]);
    printf("P%d AT: ", i);
    scanf("%d", &at[i]);
  }
```

```
// Sort by Burst Time (non-preemptive SJF)
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;
         temp = at[i]; at[i] = at[j]; at[j] = temp;
         temp = p[i];  p[i] = p[j];  p[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] - at[i];
   awt += wt[i] - at[i];
   atat += tat[i];
}

printf("\nProcess\tBT\tAT\tWT\tTAT\n");
for (i = 0; i < n; i++)
   printf("P%d\t%d\t%d\t%d\t%d\n", p[i], bt[i], at[i], wt[i] - at[i], tat[i]);

printf("\nAvg WT = %.2f", awt / n);
printf("\nAvg TAT = %.2f", atat / n);
```

```
    getch();
}
```

✅ **EX.NO: 3C – Round Robin Scheduling Algorithm**

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

void main() {
    int i, j, n, bt[10], rt[10], wt[10], tat[10], time = 0, tq;
    float awt = 0, atat = 0;
    clrscr();

    printf("Enter number of processes: ");
    scanf("%d", &n);
    printf("Enter Burst Time for each process:\n");
    for (i = 0; i < n; i++) {
        printf("P%d: ", i + 1);
        scanf("%d", &bt[i]);
        rt[i] = bt[i];
    }

    printf("Enter Time Quantum: ");
    scanf("%d", &tq);

    int done;
    do {
```

```
    done = 1;
    for (i = 0; i < n; i++) {
      if (rt[i] > 0) {
        done = 0;
        if (rt[i] > tq) {
          time += tq;
          rt[i] -= tq;
        } else {
          time += rt[i];
          wt[i] = time - bt[i];
          rt[i] = 0;
        }
      }
    }
  } while (!done);

  for (i = 0; i < n; i++) {
    tat[i] = bt[i] + wt[i];
    awt += wt[i];
    atat += tat[i];
  }

  printf("\nProcess\tBT\tWT\tTAT\n");
  for (i = 0; i < n; i++)
    printf("P%d\t%d\t%d\t%d\n", i + 1, bt[i], wt[i], tat[i]);
```

```
    printf("\nAvg WT = %.2f", awt / n);
    printf("\nAvg TAT = %.2f", atat / n);
    getch();
}
```

✅ **EX.NO: 3D – Priority Scheduling Algorithm**

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

void main() {
    int bt[10], pr[10], wt[10], tat[10], p[10], i, j, n, temp;
    float awt = 0, atat = 0;
    clrscr();

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

    for (i = 0; i < n; i++) {
        printf("P%d Burst Time: ", i + 1);
        scanf("%d", &bt[i]);
        printf("P%d Priority (lower = higher priority): ", i + 1);
        scanf("%d", &pr[i]);
        p[i] = i + 1;
    }

    // Sort by priority
```

```
  for (i = 0; i < n - 1; i++)
     for (j = i + 1; j < n; j++)
        if (pr[i] > pr[j]) {
           temp = pr[i]; pr[i] = pr[j]; pr[j] = temp;
           temp = bt[i]; bt[i] = bt[j]; bt[j] = temp;
           temp = p[i];  p[i] = p[j];  p[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("\nProcess\tPriority\tBT\tWT\tTAT\n");
  for (i = 0; i < n; i++)
     printf("P%d\t%d\t\t%d\t%d\t%d\n", p[i], pr[i], bt[i], wt[i], tat[i]);

  printf("\nAvg WT = %.2f", awt / n);
  printf("\nAvg TAT = %.2f", atat / n);
  getch();
}
```

✅ **EX.NO: 4 – Dining Philosophers (Simulated without Threads/Semaphores)**

c

CopyEdit

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

void main() {
  int i, n, hungry[10], h, ch, j;

  clrscr();
  printf("DINING PHILOSOPHER PROBLEM\n");
  printf("Enter number of philosophers: ");
  scanf("%d", &n);

  printf("How many are hungry: ");
  scanf("%d", &h);

  for (i = 0; i < h; i++) {
    printf("Enter philosopher %d position (0 to %d): ", i + 1, n - 1);
    scanf("%d", &hungry[i]);
  }

  do {
```

```
    printf("\n1. One can eat at a time\n2. Two can eat at a time\n3. Exit\nEnter your choice:
");
    scanf("%d", &ch);

    if (ch == 1) {
      for (i = 0; i < h; i++) {
        printf("\nPhilosopher %d is EATING", hungry[i]);
        for (j = 0; j < h; j++) {
          if (j != i) printf("\nPhilosopher %d is WAITING", hungry[j]);
        }
      }
    } else if (ch == 2) {
      for (i = 0; i < h - 1; i++) {
        if ((hungry[i] + 1) % n != hungry[i + 1]) {
          printf("\nPhilosopher %d and %d are EATING", hungry[i], hungry[i + 1]);
          for (j = 0; j < h; j++) {
            if (j != i && j != i + 1)
              printf("\nPhilosopher %d is WAITING", hungry[j]);
          }
        }
      }
    } else if (ch != 3) {
      printf("Invalid choice!");
    }
  } while (ch != 3);
  getch();
```

```
}
```

✅ **EX.NO: 5 – Banker's Algorithm (Deadlock Avoidance)**

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

void main() {
   int alloc[10][10], max[10][10], avail[10], need[10][10];
   int p, r, i, j, finish[10] = {0}, safe[10], k = 0;

   clrscr();
   printf("Enter number of processes and resources: ");
   scanf("%d%d", &p, &r);

   printf("Enter allocation matrix:\n");
   for (i = 0; i < p; i++)
      for (j = 0; j < r; j++)
         scanf("%d", &alloc[i][j]);

   printf("Enter max matrix:\n");
   for (i = 0; i < p; i++)
      for (j = 0; j < r; j++)
         scanf("%d", &max[i][j]);

   printf("Enter available resources:\n");
   for (i = 0; i < r; i++)
```

```
    scanf("%d", &avail[i]);

for (i = 0; i < p; i++)
    for (j = 0; j < r; j++)
        need[i][j] = max[i][j] - alloc[i][j];

int flag = 1;
while (flag) {
    flag = 0;
    for (i = 0; i < p; i++) {
        if (!finish[i]) {
            int possible = 1;
            for (j = 0; j < r; j++)
                if (need[i][j] > avail[j])
                    possible = 0;

            if (possible) {
                for (j = 0; j < r; j++)
                    avail[j] += alloc[i][j];
                finish[i] = 1;
                safe[k++] = i;
                flag = 1;
            }
        }
    }
}
```

```
  for (i = 0; i < p; i++) {
    if (!finish[i]) {
      printf("\nSystem is in UNSAFE state.");
      getch();
      return;
    }
  }

  printf("\nSystem is in SAFE state.\nSafe Sequence: ");
  for (i = 0; i < p; i++)
    printf("P%d ", safe[i]);
  getch();
}
```

✅ **EX.NO: 5B – Deadlock Detection Algorithm**

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

void main() {
  int alloc[10][10], max[10][10], avail[10], need[10][10], finish[10] = {0};
  int p, r, i, j, work[10], count = 0;
  clrscr();

  printf("Enter number of processes and resources: ");
  scanf("%d%d", &p, &r);
```

```
printf("Enter allocation matrix:\n");
for (i = 0; i < p; i++)
   for (j = 0; j < r; j++)
      scanf("%d", &alloc[i][j]);

printf("Enter max matrix:\n");
for (i = 0; i < p; i++)
   for (j = 0; j < r; j++)
      scanf("%d", &max[i][j]);

printf("Enter available resources:\n");
for (i = 0; i < r; i++) {
   scanf("%d", &avail[i]);
   work[i] = avail[i];
}

for (i = 0; i < p; i++)
   for (j = 0; j < r; j++)
      need[i][j] = max[i][j] - alloc[i][j];

for (i = 0; i < p; i++) {
   int can_run = 1;
   for (j = 0; j < r; j++)
      if (need[i][j] > work[j])
         can_run = 0;
```

```
      if (can_run) {
         for (j = 0; j < r; j++)
            work[j] += alloc[i][j];
         finish[i] = 1;
         count++;
      }
   }

   if (count == p)
      printf("No deadlock detected.");
   else {
      printf("Deadlock detected. Processes:");
      for (i = 0; i < p; i++)
         if (!finish[i]) printf(" P%d", i);
   }

   getch();
}
```

✅ **EX.NO: 6 – Paging Technique**

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

void main() {
   int ms, ps, nop, np, i, j, rempages, fno[10][10], s[10];
```

```
int x, y, offset, pa;

clrscr();
printf("Enter memory size: ");
scanf("%d", &ms);
printf("Enter page size: ");
scanf("%d", &ps);

nop = ms / ps;
printf("Number of pages available = %d\n", nop);

printf("Enter number of processes: ");
scanf("%d", &np);
rempages = nop;

for (i = 0; i < np; i++) {
   printf("Pages required for P[%d]: ", i + 1);
   scanf("%d", &s[i]);
   if (s[i] > rempages) {
      printf("Memory full!\n");
      break;
   }
   rempages -= s[i];
   printf("Enter page table for P[%d]: ", i + 1);
   for (j = 0; j < s[i]; j++)
      scanf("%d", &fno[i][j]);
```

```
    }

    printf("Enter process no., page no. and offset: ");
    scanf("%d %d %d", &x, &y, &offset);

    if (x > np || y >= s[x - 1] || offset >= ps)
        printf("Invalid input!\n");
    else {
        pa = fno[x - 1][y] * ps + offset;
        printf("Physical Address = %d\n", pa);
    }

    getch();
}
```

✅ **EX.NO: 7A – FIFO Page Replacement**

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

void main() {
    int i, j, n, f, rs[30], frame[10], count = 0, pf = 0;
    clrscr();

    printf("Enter length of reference string: ");
    scanf("%d", &n);
    printf("Enter reference string: ");
```

```
for (i = 0; i < n; i++)
   scanf("%d", &rs[i]);

printf("Enter number of frames: ");
scanf("%d", &f);

for (i = 0; i < f; i++)
   frame[i] = -1;

for (i = 0; i < n; i++) {
   int flag = 0;
   for (j = 0; j < f; j++)
      if (frame[j] == rs[i])
         flag = 1;

   if (!flag) {
      frame[count] = rs[i];
      count = (count + 1) % f;
      pf++;
   }

   for (j = 0; j < f; j++)
      printf("%d\t", frame[j]);
   if (!flag) printf("Page Fault %d", pf);
   printf("\n");
}
```

```
    printf("\nTotal Page Faults = %d", pf);
    getch();
}
```

✅ **EX.NO: 7B – LRU Page Replacement**

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

void main() {
    int i, j, n, f, rs[30], frame[10], time[10], t = 0, pf = 0;
    clrscr();

    printf("Enter length of reference string: ");
    scanf("%d", &n);
    printf("Enter reference string: ");
    for (i = 0; i < n; i++)
        scanf("%d", &rs[i]);

    printf("Enter number of frames: ");
    scanf("%d", &f);

    for (i = 0; i < f; i++) {
        frame[i] = -1;
        time[i] = 0;
```

```
}

for (i = 0; i < n; i++) {
   int flag = 0;
   for (j = 0; j < f; j++) {
      if (frame[j] == rs[i]) {
         flag = 1;
         time[j] = ++t;
      }
   }

   if (!flag) {
      int min = 0;
      for (j = 1; j < f; j++)
         if (time[j] < time[min])
            min = j;
      frame[min] = rs[i];
      time[min] = ++t;
      pf++;
   }

   for (j = 0; j < f; j++)
      printf("%d\t", frame[j]);
   if (!flag) printf("Page Fault %d", pf);
   printf("\n");
}
```

```
    printf("\nTotal Page Faults = %d", pf);
    getch();
}
```

✅ **EX.NO: 7C – Optimal Page Replacement (Approximated in TC)**

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

int predict(int pages[], int n, int frames[], int f, int index) {
    int res = -1, farthest = index, i, j;
    for (i = 0; i < f; i++) {
        for (j = index; j < n; j++) {
            if (frames[i] == pages[j]) {
                if (j > farthest) {
                    farthest = j;
                    res = i;
                }
                break;
            }
        }
        if (j == n) return i;
    }
    return (res == -1) ? 0 : res;
}
```

```
void main() {
   int pages[30], frames[10], n, f, i, j, pf = 0;
   clrscr();

   printf("Enter number of pages: ");
   scanf("%d", &n);
   printf("Enter reference string: ");
   for (i = 0; i < n; i++)
      scanf("%d", &pages[i]);

   printf("Enter number of frames: ");
   scanf("%d", &f);
   for (i = 0; i < f; i++)
      frames[i] = -1;

   for (i = 0; i < n; i++) {
      int flag = 0;
      for (j = 0; j < f; j++) {
         if (frames[j] == pages[i]) {
            flag = 1;
            break;
         }
      }

      if (!flag) {
         int pos = (i < f) ? i : predict(pages, n, frames, f, i + 1);
```

```
            frames[pos] = pages[i];
            pf++;
        }

        for (j = 0; j < f; j++)
            printf("%d\t", frames[j]);
        if (!flag) printf("Page Fault %d", pf);
        printf("\n");
    }

    printf("\nTotal Page Faults = %d", pf);
    getch();
}
```

✅ **EX.NO: 8 – Disk Scheduling (FCFS Only, SSTF & SCAN next)**

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

void main() {
    int n, i, req[20], head, seek = 0;
    clrscr();

    printf("Enter number of requests: ");
    scanf("%d", &n);
```

```
    printf("Enter request queue: ");
    for (i = 0; i < n; i++)
        scanf("%d", &req[i]);

    printf("Enter initial head position: ");
    scanf("%d", &head);

    for (i = 0; i < n; i++) {
        seek += abs(req[i] - head);
        head = req[i];
    }

    printf("Total Seek Time: %d", seek);
    getch();
}
```

✅ **EX.NO: 8 (continued) – SSTF Disk Scheduling**

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

void main() {
    int n, i, j, head, pos, diff, min, done[20] = {0}, req[20], seek = 0;
    clrscr();

    printf("Enter number of requests: ");
```

```
scanf("%d", &n);

printf("Enter request queue: ");
for (i = 0; i < n; i++)
   scanf("%d", &req[i]);

printf("Enter initial head position: ");
scanf("%d", &head);

for (i = 0; i < n; i++) {
   min = 9999;
   for (j = 0; j < n; j++) {
      if (!done[j]) {
         diff = abs(head - req[j]);
         if (diff < min) {
            min = diff;
            pos = j;
         }
      }
   }
   seek += min;
   head = req[pos];
   done[pos] = 1;
}

printf("Total Seek Time: %d", seek);
```

```
   getch();
}
```

✅ **EX.NO: 8 (continued) – SCAN Disk Scheduling (Upwards Direction)**

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

void sort(int a[], int n) {
   int i, j, t;
   for (i = 0; i < n - 1; i++)
      for (j = i + 1; j < n; j++)
         if (a[i] > a[j]) {
            t = a[i];
            a[i] = a[j];
            a[j] = t;
         }
}

void main() {
   int n, i, head, req[20], max, seek = 0;
   clrscr();

   printf("Enter max disk size: ");
   scanf("%d", &max);
```

```
printf("Enter number of requests: ");
scanf("%d", &n);

printf("Enter request queue: ");
for (i = 0; i < n; i++)
   scanf("%d", &req[i]);

printf("Enter initial head position: ");
scanf("%d", &head);

req[n] = head;
n++;
sort(req, n);

int pos;
for (i = 0; i < n; i++)
   if (req[i] == head)
      pos = i;

for (i = pos; i < n - 1; i++) {
   seek += abs(req[i + 1] - req[i]);
}
seek += abs(max - 1 - req[n - 1]);

for (i = pos - 1; i >= 0; i--) {
   if (i == pos - 1)
```

```
            seek += abs(max - 1 - req[i]);
        else
            seek += abs(req[i + 1] - req[i]);
    }

    printf("Total Seek Time: %d", seek);
    getch();
}
```