Program: Priority Scheduling Without Arrival Time.

CODE:

```
#include #include using namespace std;

struct Process { int id, bt, priority, wt, tat; };

// Comparator function to sort by priority (lower value = higher
priority) bool compare(Process a, Process b) { return a.priority <
b.priority; }

void priorityScheduling(Process p[], int n) { sort(p, p + n, compare);
// Sort based on priority

p[0].wt = 0;  // First process has no waiting time
p[0].tat = p[0].bt;

float total_wt = 0, total_tat = p[0].tat;

for (int i = 1; i < n; i++) {
    p[i].wt = p[i - 1].wt + p[i - 1].bt;
    p[i].tat = p[i].wt + p[i].bt;

    total_wt += p[i].wt;
    total_tat += p[i].tat;
}

cout << "\nProcess\tBT\tPriority\tWT\tTAT\n";
for (int i = 0; i < n; i++) {
    cout << p[i].id << "\t" << p[i].bt << "\t" << p[i].priority <<
"\t\t"
         << p[i].wt << "\t" << p[i].tat << endl;
}

cout << "\nAverage Waiting Time: " << (total_wt / n);
cout << "\nAverage Turnaround Time: " << (total_tat / n) << endl;


}
```

```
int main() { int n; cout << "Enter number of processes: "; cin >> n;

Process p[n];

for (int i = 0; i < n; i++) {
    cout << "Enter burst time and priority for process " << i + 1 <<
": ";
    cin >> p[i].bt >> p[i].priority;
    p[i].id = i + 1;
}

priorityScheduling(p, n);
return 0;
 }
```

OUTPUT:

```
Enter number of processes: 3

Enter burst time and priority for process 1: 5 2

Enter burst time and priority for process 2: 3 1

Enter burst time and priority for process 3: 8 3


Process    BT    Priority    WT    TAT

2         3     1           0     3

1         5     2           3     8

3         8     3           8     16


Average Waiting Time: 3.66667

Average Turnaround Time: 9



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