



# **DATA STRUCTURES PROJECT BATCH 10**

**BY:**

**1601-21-733-071    Srija.B**  
**1601-71-733-072    Chandana.B**  
**1601-21-733-081    Aishwarya.N**  
**1601-21-733-085    Swathi.S**  
**1601-21-733-089    Manasa.V**  
**1601-21-733-118    Aditya Ram**

**PROBLEM STATEMENT:**

Alice has two queues, Q and R, which can store integers. Bob gives Alice 50 odd integers and 50 even integers and insists that she store all 100 integers in Q and R. They then play a game where Bob picks Q or R at random and then applies the round-robin scheduler, described in the chapter, to the chosen queue a random number of times. If the last number to be processed at the end of this game was odd, Bob wins. Otherwise, Alice wins. How can Alice allocate integers to queues to optimize her chances of winning? What is her chance of winning?

## ALGORITHM:

Since linked list is used, each node is a process, which requires unique burst time, completion time so are created in struct node struct node{

```
    int data;    int
    bursttime;  int
    initial_bursttime;  int
    completion_time;  struct
    node *link;
};
```

```
main(){
    numbers();
    create();    assign();
    roundrobin();
}
```

```
numbers(){ create array_e using random function containing 50 even
numbers    create array_o using random function containing 50 odd
numbers    combine array_e and array_o
}
```

```
create(){
    input(length of queue Q)
    length of R=100-length of Q
    create_queue(length of Q)
    create_queue(length of R)
}
create_queue(){
    create Q of length_q using linked list

    create R of length_r using linked list
```

```

}

assign(){
    input( Enter numbers from array given by bob)
    store numbers in queue Q
    store remaining numbers in R
}

roundrobin(){
    input(Select a queue)
    while( end of queue ){
        input(Enter burst time for each number in queue)
        store initial_burst time
        initiate completion time to 0
    }
    input(Enter quantum time)
    temp=first element in queue
    while(){
        if(bursttime_temp > quantumtime){
            -->increment completion_time_array
            -->(bursttime_temp)=(bursttime_temp)-quantum_time
            -->increment final_order_array
            -->temp=temp->link;
        }
        if(0 <bursttime_temp <= quantumtime){
            -->increment completion_time_array
            -->print(completion and waiting time of that particular
process)
            -->store the completion and waiting time of that process
calculate avg    to
                  completion and waiting time
            -->delete that node from queue
        }
        if( end of queue reached){
            return to starting of queue
        }
    }
}

```

}

## DESCRIPTION:

Two arrays, array\_e and array\_o are created, containing 50 even random numbers and 50 odd numbers, which are given by Bob. Length of queue Q is taken as input, and length of queue R is 100-(length of queue). The queues are created using linked lists and elements from the combined array (even and odd) are added into Q and remaining numbers are added into R.

The node in the linked lists should contain data, burst\_time ,initial\_burst\_time, completion time.

Round Robin scheduling is applied to the selected queue. Quantum time is taken as input, and each and every element in the node is executed till the node's burst time is over, and each time, the element is noted in the final array. And once the burst time is over, the node is deleted from the queue. The completion (turn around time) and waiting time for each node is calculated. This process is continued till all the nodes in the queue are deleted.

We get the final array, average completion time and average waiting time.

If the number in the final array is even, Alice wins, else Bob wins.

## CODE:

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

```
int array_e[52],array_o[52],array_combine[104];
```

```
struct node{    //creating a node with elements- data,burst time,initial burst  
time,completion time
```

```
    int data;
```

```
    int bt;        int
```

```
i_bt; int ct;
```

```
struct node *link; };
```

```
typedef struct node node;
```

```

node *front_q=NULL,*rear_q=NULL,*front_r=NULL,*rear_r=NULL; int
length_q,length_r;

void create_queue(int length,node *front,node *rear){

    node *newnode;

    newnode=(node*)malloc(sizeof(node));

    newnode->data=0;        int temp=0;

    while(temp<length){
        //CREATING
        QUEUES OF REQUIRED LENGTH

        if (temp==0){

            front=rear=newnode;

            front->link=NULL;

            rear->link=NULL;

        }

        else{

            newnode=(node*)malloc(sizeof(node));

            rear->link=newnode;

            rear=newnode;        rear->link=NULL;

        }

        temp++;

    }

    if(length==length_r){

        front_r=front;

        rear_r=rear;

    }

```

```

        else if(length==length_q){
            front_q=front;
rear_q=rear;
        }
    }

void create(){
    printf("\n\nEnter the size of queue Q \n\n");
    scanf("%d",&length_q); length_r=100-
length_q;
    create_queue(length_q,front_q,rear_q); //CREATING TWO QUEUES
    create_queue(length_r,front_r,rear_r);

} void
numbers(){
    int i=0,j=0,k=0,num;    //STORING RANDOM EVEN
NUMBERS AND RANDOM ODD NUMBERS IN TWO DIFFERENT
ARRAYS    while(i<50 || j<50){        num=rand();
if(num%2==0 && i<50){
            array_e[i]=num;
            i++;

        }
    }
}

```

```

        else if(num%2!=0 && j<50){

array_o[j]=num;

        j++;

        }

    }

    printf("\nList of even integers \n");

    for(i=0;i<50;i++){                                //COMBINING EVEN
AND ODD ARRAY INTO SINGLE ARRAY

array_combine[k]=array_e[i];                        printf("%d.%d
",i+1,array_e[i]);

        k++;

    }

    printf("\n\n");

    printf("\nList of odd integers \n");

    for(j=0;j<50;j++){

array_combine[k]=array_o[j];                        printf("%d. %d
",j+1,array_o[j]);

        k++;

    }

    printf("\n\n");

    printf("\nThe 100 integers Bob gave are\n");

    for(k=0;k<100;k++){

        printf("%d. %d ",k+1,array_combine[k]);

    }

```

```

} void assign(){
int a,i,flag=0,k;
node *temp;

    printf("Alice !Please assign numbers in Q from the list BOB gave!!\n");
temp=front_q;    while(temp!=rear_q->link){

        scanf("%d",&a);                                //ALICE ENTERS
DATA IN QUEUE Q..

        temp->data=a;
temp=temp->link;

    }

    printf("The elements in queue Q are: \n"); node
*temp1,*temp2;                temp1=front_q;
while(temp1!=rear_q->link){    printf("%d
",temp1->data); temp1=temp1->link;

    }

    printf("\n\n");

    printf("The elements in queue R are \n"); //REMAINING ELEMENTS IN
THE QUEUE R ARE ADDED..

    temp1=front_q;
    //COMBINED ARRAY CONSISTING OF EVEN AND ODD
NUMBERS ARE COMPARED WITH

    temp2=front_r;
    //ELEMENTS IN Q, IF THEY ARE NOT SAME, THE ELEMENT IS

```



ADDED IN QUEUE R

```
int                iter;

for(i=0;i<100;i++){
    flag=0;

    temp1=front_q;
    iter=array_combine[i];
    while(temp1!=rear_q->link){
        if(temp1->data==iter){
            flag=1;

            break;
        }
        temp1=temp1->link;
    }
    if(flag==0){
        temp2->data=iter;
        printf("%d ",temp2->data);
        temp2=temp2->link;
    }
}

printf("\n\n");

void roundrobin(){
    printf("BOB! pick a queue!\n");
    int
```

```
i=0,j=0,k=0,qt,count=0,sq=0,final[100],ct[100],wt[100],sum=0,temp1,len,sum1=0; //finalorder array,
```

```
node *temp,*iter,*front,*rear;
```

```
//completion time array,waiting time array,quantum time,length..
```

```
char c; //of array(Q or
```

```
R)
```

```
scanf("%s",&c);
```

```
if(c=='Q'){
```

```
front=front_q;
```

```
rear=rear_q;
```

```
len=length_q;
```

```
}
```

```
else if(c=='R'){
```

```
front=front_r;
```

```
rear=rear_r;
```

```
len=length_r;
```

```
}
```

```
temp=front;
```

```
while(temp!=rear->link){
```

```
printf("Enter the burst time for process %d ",temp->data);
```

```
scanf("%d",&temp->bt);
```

```
temp->i_bt=temp->bt; //STORING INITIAL  
BURST TIME
```

```
temp->ct=0; //INITIATING  
COMPLETION TIME TO 0
```

```
temp=temp->link;
```

```

    }

    printf("Enter the quantum time");          //EACH TIME
    WE ENCOUNTER A PROCESS, WE ADD IT TO FINAL ARRAY
    scanf("%d",&qt); temp=front;      ct[0]=0;    wt[0]=0;
    while(temp!=NULL){
        if((temp->bt)>qt){                  //IF REMAINING BURST
        TIME OF THE PROCESS IS GREATER THAN QUANTUM
            temp1=ct[count];                //TIME,FIRST, WE
        INCREMENT THE CT ARRAY,THEN REMANINING BURST TIME
            count++;                        //CALCULATED..
            ct[count]=temp1+qt;              (temp-
        >bt)=(temp->bt)-qt;                    final[i]=temp->data;
            temp=temp->link;                  //PROCESS EXECUTION
        DONE, WE PROCEES TO NEXT NODE
            if(temp==NULL){                  //IF WE REACH
        LAST NODE, WE AGAIN START FROM FIRST
                temp=front;
            }
            i++;
            continue;
        }
        else if((temp->bt)==qt || 0<=(temp->bt)<qt){
            temp1=ct[count];
count++;
            ct[count]=temp1+temp->bt;
            printf("\nThe completion time of %d is %d\n",temp-

```

```

>data,ct[count]);

        printf("\nThe waiting time of %d is %d\n",temp-
>data,ct[count]-temp->i_bt);

        sum1=sum1+ct[count]-temp->i_bt;           //IF BURST
TIME ID LESS THAN OR EQUAL TO QUANTUM TIME,

        sum=sum+ct[count];                         //THE
COMPLETION AND WAITING TIME OF THE PROCESS IS RECORDED
        final[i]=temp->data;                       //AND THE
PROCESS IS REMOVED FROM THE QUEUE

        iter=front;

        if(temp==front){                           //REMOVAL IF
PROCESS IS FRONT NODE
                temp=temp->link;

        front=temp;

        }

        else{

                while(iter!=temp->link)

                {

                        if(iter->link==temp){

                                iter->link=temp->link;           //REMOVAL
IF PROCESS IS OTHER THAN FRONT NODE

                                free(temp);

                                temp=iter->link;

                                break;

                        }

                        iter=iter->link;

                }

```

```

        }

        i++;

        if(temp==NULL){
            temp=front;
        }
    }

}

    printf("The final array is");    //FINAL ARRAY CONSISTING OF
    THE ORDER OF THE PROCESSES IS DISPLAYED

    for(k=0;k<i;k++){    printf("%d ",final[k]);

        }

    printf("\n\nThe avg completion time is %d/%d \n",sum,len);

    printf("\n\nThe avg waiting time is %d/%d \n",sum1,len); if(final[i-1]%2!=0){

        printf("\n\n BOB WON!!Sorry ALICE! better luck next time!!");
        //IF THE LAST ELEMENT IN THE FINAL ARRAY IS
        ODD, BOB WINS ELSE, ALICE WINS

        }

        else{

            printf("\n\nALICE WON!! Sorry BOB better luck next time!!");

        }

    }
}

```

```
void main(){  
    printf("-----START GAME-----"  
-----");  
    printf("\n\n");  
    numbers(); create();  
    assign();  
    roundrobin();  
  
}
```

**OUTPUT:**

```

-----START GAME-----

List of even integers
1.6334 2.26500 3.15724 4.11478 5.29358 6.26962 7.24464 8.11942 9.5436 10.14604 11.3902 12.292 13.12382 14.18716 15.19718 16.21726 17.11538 18.19912 19.9894 20.31322 21.4664
22.6868 23.27644 24.32662 25.778 26.12316 27.22190 28.1842 29.288 30.30106 31.9040 32.8942 33.19264 34.22648 35.27446 36.15890 37.24370 38.15350 39.15006 40.3548 41.24084
42.19954 43.18756 44.11840 45.4966 46.7376 47.26308 48.16944 49.24626 50.21538

List of odd integers
1. 41 2. 18467 3. 19169 4. 5705 5. 28145 6. 23281 7. 16827 8. 9961 9. 491 10. 2995 11. 4827 12. 32391 13. 153 14. 17421 15. 19895 16. 5447 17. 14771 18. 1869 19. 25667 20.
26299 21. 17035 22. 28703 23. 23811 24. 30333 25. 17673 26. 15141 27. 7711 28. 28253 29. 25547 30. 32757 31. 20037 32. 12859 33. 8723 34. 9741 35. 27529 36. 3035 37. 23805
38. 6729 39. 31101 40. 24393 41. 19629 42. 12623 43. 13931 44. 32439 45. 11323 46. 5537 47. 22929 48. 16541 49. 4833 50. 31115

The 100 integers Bob gave are
1. 6334 2. 26500 3. 15724 4. 11478 5. 29358 6. 26962 7. 24464 8. 11942 9. 5436 10. 14604 11. 3902 12. 292 13. 12382 14. 18716 15. 19718 16. 21726 17. 11538 18. 19912 19. 98
94 20. 31322 21. 4664 22. 6868 23. 27644 24. 32662 25. 778 26. 12316 27. 22190 28. 1842 29. 288 30. 30106 31. 9040 32. 8942 33. 19264 34. 22648 35. 27446 36. 15890 37. 2437
0 38. 15350 39. 15006 40. 3548 41. 24084 42. 19954 43. 18756 44. 11840 45. 4966 46. 7376 47. 26308 48. 16944 49. 24626 50. 21538 51. 41 52. 18467 53. 19169 54. 5705 55. 281
45 56. 23281 57. 16827 58. 9961 59. 491 60. 2995 61. 4827 62. 32391 63. 153 64. 17421 65. 19895 66. 5447 67. 14771 68. 1869 69. 25667 70. 26299 71. 17035 72. 28703 73. 2381
1 74. 30333 75. 17673 76. 15141 77. 7711 78. 28253 79. 25547 80. 32757 81. 20037 82. 12859 83. 8723 84. 9741 85. 27529 86. 3035 87. 23805 88. 6729 89. 31101 90. 24393 91. 1
9629 92. 12623 93. 13931 94. 32439 95. 11323 96. 5537 97. 22929 98. 16541 99. 4833 100. 31115

Enter the size of queue Q
8
Alice !Please assign numbers in Q from the list BOB gave!!
6334
31115
11942
3902
26500
7711
4827
153
The elements in queue Q are:
6334 31115 11942 3902 26500 7711 4827 153
The elements in queue R are
15724 11478 29358 26962 24464 5436 14604 292 12382 18716 19718 21726 11538 19912 9894 31322 4664 6868 27644 32662 778 12316 22190 1842 288 30106 9
040 8942 19264 22648 27446 15890 24370 15350 15006 3548 24084 19954 18756 11840 4966 7376 26308 16944 24626 21538 41 18467 19169 5705 28145 23281
16827 9961 491 2995 32391 17421 19895 5447 14771 1869 25667 26299 17035 28703 23811 30333 17673 15141 28253 25547 32757 20037 12859 8723 9741 27529
3035 23805 6729 31101 24393 19629 12623 13931 32439 11323 5537 22929 16541 4833

BOB! pick a queue!
Q
Enter the burst time for process 6334 5
Enter the burst time for process 31115 3
Enter the burst time for process 11942 8
Enter the burst time for process 3902 6
Enter the burst time for process 26500 2
Enter the burst time for process 7711 9
Enter the burst time for process 4827 4
Enter the burst time for process 153 7
Enter the quantum time3

The completion time of 31115 is 6

The waiting time of 31115 is 3

The completion time of 26500 is 14

The waiting time of 26500 is 12

The completion time of 6334 is 25

The waiting time of 6334 is 20

The completion time of 3902 is 31

The waiting time of 3902 is 25

The completion time of 4827 is 35

The waiting time of 4827 is 31

The completion time of 11942 is 40

The waiting time of 11942 is 32

The completion time of 7711 is 43

The waiting time of 7711 is 34

The completion time of 153 is 44

The waiting time of 153 is 37
The final array is6334 31115 11942 3902 26500 7711 4827 153 6334 11942 3902 7711 4827 153 11942 7711 153

The avg completion time is 238/8

The avg waiting time is 194/8

BOB WON!!Sorry ALICE! better luck next time!!
-----
Process exited after 111.4 seconds with return value 48
Press any key to continue . . .

```

How can Alice allocate integers to queues to optimize her chances of winning?

What is her chance of winning?

## TEST CASE:

Alice should put on an even integer in Q and all the other 99 integers in R.

The chance that Bob picks Q is 0.5, where Alice must win.

The chance that Bob picks R is also 0.5, where Alice wins at the chance of 49/99 (49 even integers out of 99 overall integers).

This gives her  $0.5 \cdot (1 + 49/99) = 74/99$  chance of winning.

## ALANYSIS:

Time complexity of best case is  $O(1)$ , which is the test case.

Average time complexity is  $O(\text{burst\_time}/\text{quantum\_time})$  if the burst times of all the processes are taken to be same.

## SOLUTION:

```
List of even integers
1. 6334 2.26580 3.15724 4.11478 5.29358 6.26962 7.24464 8.11942 9.5436 10.14604 11.3902 12.292 13.12382 14.18716 15.19718 16.21726 17.11538 18.19912 19.9894 20.31322 21.4664 22.6868 23.2764
4. 24.22662 25.778 26.12316 27.22190 28.1842 29.288 30.30106 31.9040 32.8942 33.19264 34.22648 35.27446 36.15890 37.24370 38.15350 39.15006 40.3548 41.24884 42.19954 43.18756 44.11848 45.49
66 46.7376 47.26308 48.16944 49.24626 50.21538

List of odd integers
1. 41 2. 18467 3. 19169 4. 5705 5. 28145 6. 23281 7. 16827 8. 9961 9. 491 10. 2995 11. 4827 12. 32391 13. 153 14. 17421 15. 19895 16. 5447 17. 14771 18. 1869 19. 25667 20. 26299 21. 17035
22. 28783 23. 23811 24. 30333 25. 17673 26. 15141 27. 7711 28. 28253 29. 25547 30. 32757 31. 20037 32. 12859 33. 8723 34. 9741 35. 27529 36. 3035 37. 23805 38. 6729 39. 31101 40. 24393 41.
19629 42. 12623 43. 13931 44. 32439 45. 11323 46. 5537 47. 22929 48. 16541 49. 4833 50. 31115

The 100 integers Bob gave are
1. 6334 2. 26500 3. 15724 4. 11478 5. 29358 6. 26962 7. 24464 8. 11942 9. 5436 10. 14604 11. 3902 12. 292 13. 12382 14. 18716 15. 19718 16. 21726 17. 11538 18. 19912 19. 9894 20. 31322 21.
4664 22. 6868 23. 27644 24. 32662 25. 778 26. 12316 27. 22190 28. 1842 29. 288 30. 30106 31. 9040 32. 8942 33. 19264 34. 22648 35. 27446 36. 15890 37. 24370 38. 15350 39. 15006 40. 3548 4
1. 24884 42. 19954 43. 18756 44. 11848 45. 4966 46. 7376 47. 26308 48. 16944 49. 24626 50. 21538 51. 41 52. 18467 53. 19169 54. 5705 55. 28145 56. 23281 57. 16827 58. 9961 59. 491 60. 2995
61. 4827 62. 32391 63. 153 64. 17421 65. 19895 66. 5447 67. 14771 68. 1869 69. 25667 70. 26299 71. 17035 72. 28783 73. 23811 74. 30333 75. 17673 76. 15141 77. 7711 78. 28253 79. 25547 80.
32757 81. 20037 82. 12859 83. 8723 84. 9741 85. 27529 86. 3035 87. 23805 88. 6729 89. 31101 90. 24393 91. 19629 92. 12623 93. 13931 94. 32439 95. 11323 96. 5537 97. 22929 98. 16541 99. 48
33 100. 31115

Enter the size of queue Q
1
Alice !Please assign numbers in Q from the list 808 gave!!
24464
The elements in queue Q are:
24464

The elements in queue R are
6334 26500 15724 11478 29358 26962 11942 5436 14604 3902 292 12382 18716 19718 21726 11538 19912 9894 31322 4664 6868 27644 32662 778 12316 22190 1842 288 30106
9040 8942 19264 22648 27446 15890 24370 15350 15006 3548 24884 19954 18756 11848 4966 7376 26308 16944 24626 21538 41 18467 19169 5705 28145 23281 16827 9961
491 2995 4827 32391 153 17421 19895 5447 14771 1869 25667 26299 17035 28703 23811 30333 17673 15141 7711 28253 25547 32757 20037 12859 8723 9741 27529 3035 2380
5 6729 31101 24393 19629 12623 13931 32439 11323 5537 22929 16541 4833 31115

BOB! pick a queue!
Q
Enter the burst time for process 24464 7
Enter the quantum time3

The completion time of 24464 is 7

The waiting time of 24464 is 0
The final array is24464 24464 24464

The avg completion time is 7/1

The avg waiting time is 0/1

ALICE WON!! Sorry BOB better luck next time!!
```