

18BCB0142

David B.A.De Vieira Velho

OS LAB DA

Q1.

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <semaphore.h>
int buf[100];

int in = 0, num = 20;
int out = 0;

sem_t completed;

sem_t finished;

sem_t m;

int bsize;

int c = 0;

void *
producer ()
{
    int i, item;

    for (i = 1;; i++)
    {
        if (i % num == 0)

            sleep (10);           //change the value inside sleep
            item = c * 10 + c;

        sem_wait (&finished);

        sem_wait (&m);

        buf[in] = item;

        in = (in + 1) % (bsize);
```

```

c++;

printf ("producer produced %d\n ", item);

sem_post (&m);

sem_post (&completed);
}
}

void *
consumer ()
{

int i, item;

for (i = 1;; i++)

{

if (i % num == 0)

sleep (10);

sem_wait (&completed);

sem_wait (&m);

item = buf[out];

out = (out + 1) % (bsize);

c--;

printf ("consumer consumed %d\n ", item);

sem_post (&m);

sem_post (&finished);
}

}

int
main ()
{

```

```
int i = 0, j = 0;

pthread_t prod, cons;

printf ("\nThe Buffer Size:");

scanf ("%d", &bsize);

sem_init (&completed, 0, 0);

sem_init (&finished, 0, bsize);

sem_init (&m, 0, 1);

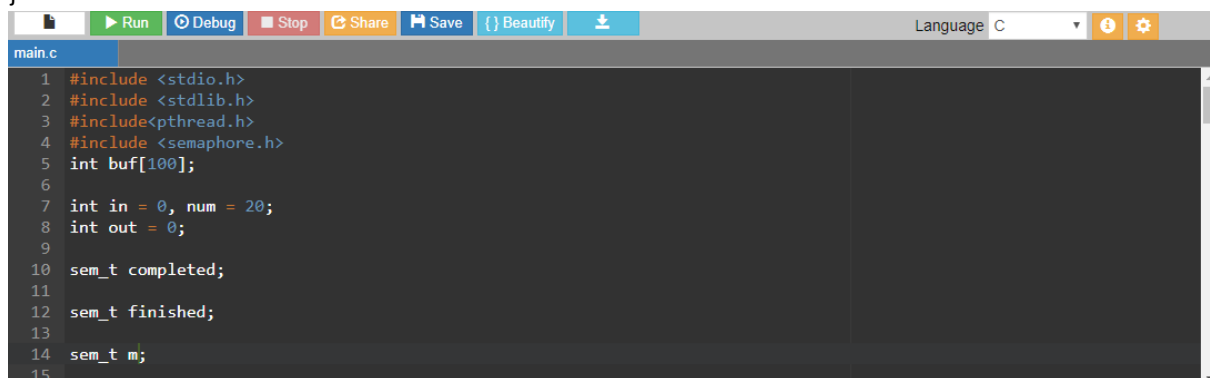
pthread_create (&prod, NULL, producer, NULL);

pthread_create (&cons, NULL, consumer, NULL);

pthread_join (prod, NULL);

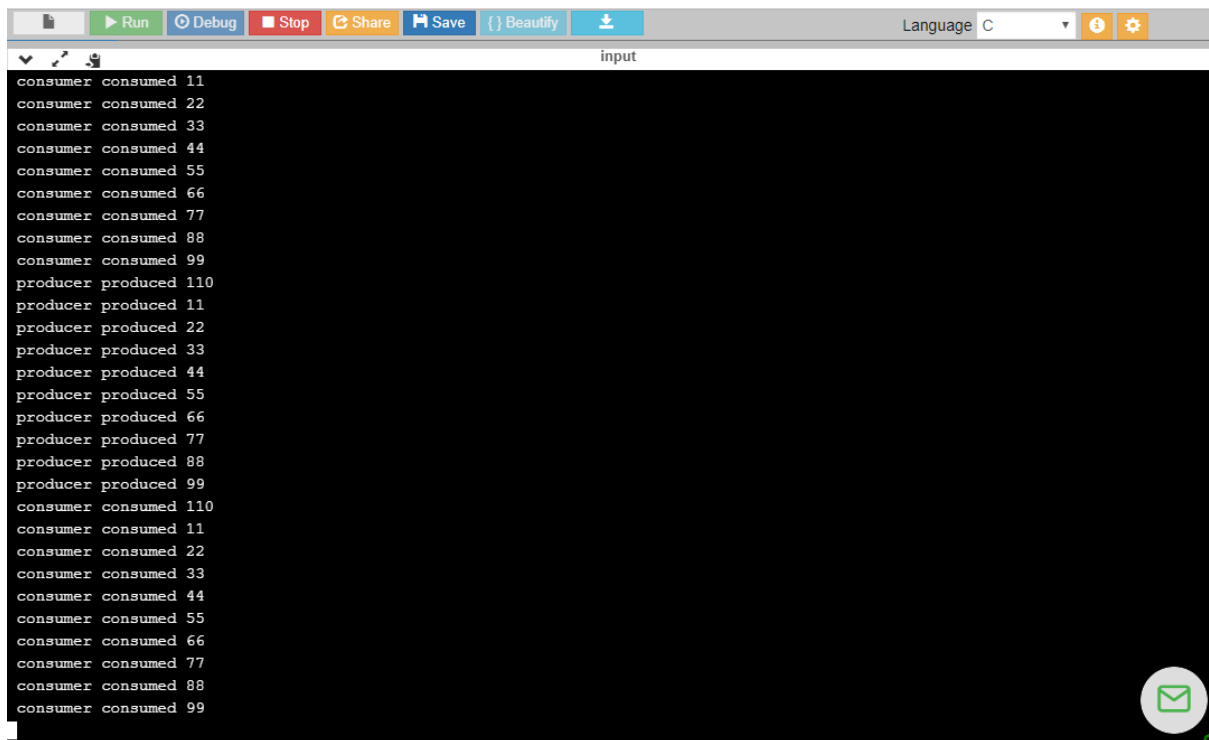
pthread_join (cons, NULL);

}
```



The screenshot shows a code editor window with a toolbar at the top containing buttons for Run, Debug, Stop, Share, Save, Beautify, and a download icon. The language is set to C. The code in the editor is as follows:

```
main.c
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include<pthread.h>
4  #include <semaphore.h>
5  int buf[100];
6
7  int in = 0, num = 20;
8  int out = 0;
9
10 sem_t completed;
11
12 sem_t finished;
13
14 sem_t m;
15
```



The screenshot shows a code editor window with a toolbar at the top containing buttons for Run, Debug, Stop, Share, Save, Beautify, and a download icon. The language is set to C. The main area displays the output of a program, with the title bar indicating the file is named 'input'. The output consists of a sequence of messages: 'consumer consumed' followed by odd numbers from 11 to 99, and 'producer produced' followed by even numbers from 110 down to 10. The messages are repeated twice. A green envelope icon is visible in the bottom right corner of the editor area.

```
consumer consumed 11
consumer consumed 22
consumer consumed 33
consumer consumed 44
consumer consumed 55
consumer consumed 66
consumer consumed 77
consumer consumed 88
consumer consumed 99
producer produced 110
producer produced 11
producer produced 22
producer produced 33
producer produced 44
producer produced 55
producer produced 66
producer produced 77
producer produced 88
producer produced 99
consumer consumed 110
consumer consumed 11
consumer consumed 22
consumer consumed 33
consumer consumed 44
consumer consumed 55
consumer consumed 66
consumer consumed 77
consumer consumed 88
consumer consumed 99
```

Q2.

```
#include<stdio.h>
struct memblock
{
    int size, flag, index;
} b[20], temp;

int n;

void
sort ()
{
    int i, j;

    struct memblock temp;

    for (i = 0; i < n - 1; i++)
    {
```

```

for (j = 0; j < n - i - 1; j++)
    {
    if (b[j].size > b[j + 1].size)
        {
        temp.size = b[j].size;
        b[j].size = b[j + 1].size;
        b[j + 1].size = temp.size;
        temp.index = b[j].index;
        b[j].index = b[j + 1].index;
        b[j + 1].index = temp.index;
        }
    }
}
}

```

```

void
revsort ()
{
    int i, j;

    struct memblock temp;

    for (i = 0; i < n - 1; i++)
        {
        for (j = 0; j < n - i - 1; j++)
            {
            if (b[j].size < b[j + 1].size)
                {
                temp.size = b[j].size;

```

```

b[j].size = b[j + 1].size;

b[j + 1].size = temp.size;

temp.index = b[j].index;

b[j].index = b[j + 1].index;

b[j + 1].index = temp.index;

}

}

}

}

```

```

int
main ()
{

int i, j, pno, p[20], flag[20], choice = 0, a;

printf ("Enter no. of memory blocks available:");

scanf ("%d", &n);

for (i = 0; i < n; i++)

{

printf ("Enter the size of %d block:", i + 1);

scanf ("%d", &b[i].size);

b[i].flag = 0;

b[i].index = i;

}

printf ("\nEnter no of processes ");

scanf ("%d", &pno);

for (i = 0; i < pno; i++)

{

```

```

printf ("Enter memory required for process %d:", i + 1);

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

flag[i] = 0;

}

while (choice != 4)

{

printf ("\nenter \n1.First fit\n2.Best fit\n3.Worst fit\n4.exit\n");

scanf ("%d", &choice);

switch (choice)

{

case 1:

for (i = 0; i < pno; i++)

{

for (j = 0; j < n; j++)

{

if (p[i] <= b[j].size && b[j].flag == 0 && flag[i] == 0)

{

printf ("\nprocess %d is allotted to memory block %d",

i + 1, b[j].index + 1);

b[j].flag = 1;

flag[i] = 1;

break;

}

}

}

if (flag[i] == 0)

printf ("\nno memory block available for process %d", i + 1);

}

```

```

for (i = 0; i < n; i++)

b[i].flag = 0;

for (i = 0; i < pno; i++)

flag[i] = 0;

break;

case 2:

sort ();

for (i = 0; i < pno; i++)

    {

for (j = 0; j < n; j++)

    {

if (p[i] <= b[j].size && b[j].flag == 0 && flag[i] == 0)

    {

printf ("\nprocess %d is allotted to memory block %d",
        i + 1, b[j].index + 1);

b[j].flag = 1;

flag[i] = 1;

    }

    }

if (flag[i] == 0)

printf ("\nno memory block available for process %d", i + 1);

    }

for (i = 0; i < n; i++)

b[i].flag = 0;

for (i = 0; i < pno; i++)

flag[i] = 0;

```



```

break;

case 3:

revsort ();

for (i = 0; i < pno; i++)
    {

for (j = 0; j < n + 1; j++)
    {

if (p[i] <= b[j].size && b[j].flag == 0 && flag[i] == 0)
    {

printf ("\nprocess %d is allotted to memory block %d",
        i + 1, b[j].index + 1);

b[j].flag = 1;

flag[i] = 1;

    }
        }

if (flag[i] == 0)

printf ("\nno memory block available for process %d", i + 1);

    }

for (i = 0; i < n; i++)

b[i].flag = 0;

for (i = 0; i < pno; i++)

flag[i] = 0;

    }

    }

printf ("\n");

return 0;

}

```

RunDebugStopShareSaveBeautify

main.c

```
131 }
132
133 while (choice != 4)
134 {
135     {
136
137     printf ("\nenter \n1.First fit\n2.Best fit\n3.Worst fit\n4.exit\n");
138
139     scanf ("%d", &choice);
```

input

```
Enter no. of memory blocks available:5
Enter the size of 1 block:2
Enter the size of 2 block:3
Enter the size of 3 block:4
Enter the size of 4 block:5
Enter the size of 5 block:6

Enter no of processes 3
Enter memory required for process 1:4
Enter memory required for process 2:5
Enter memory required for process 3:6

enter
1.First fit
2.Best fit
3.Worst fit
4.exit
```

```
Run Debug Stop Share Save {} Beautify input
1
process 1 is allotted to memory block 3
process 2 is allotted to memory block 4
process 3 is allotted to memory block 5
enter
1.First fit
2.Best fit
3.Worst fit
4.exit
2
process 1 is allotted to memory block 3
process 2 is allotted to memory block 4
process 3 is allotted to memory block 5
enter
1.First fit
2.Best fit
3.Worst fit
4.exit
3
process 1 is allotted to memory block 5
process 2 is allotted to memory block 4
no memory block available for process 3
enter
1.First fit
2.Best fit
3.Worst fit
4.exit
```