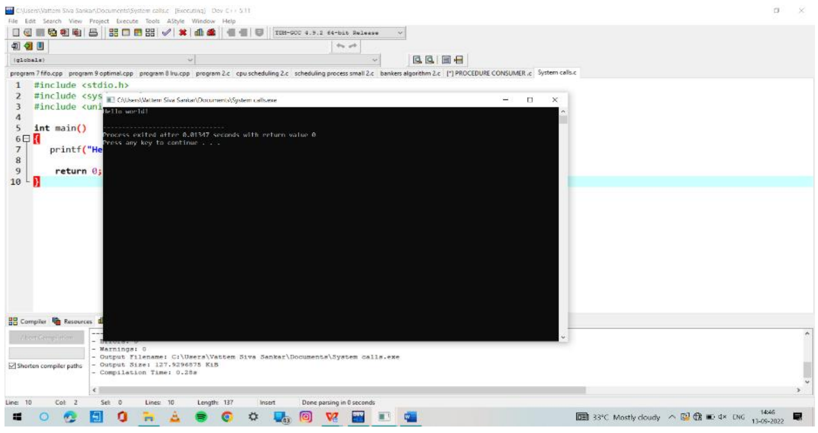
1.Create a new process by invoking the appropriate system call. Get the process identifier of the currently running process and its respective parent using system calls and display the same using a C program.

Program:

```
#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>


int main()

{

  printf("Hello world!\n");


   return 0;

}
```

Output:



```
Hello world!

--------------------------------
Process exited after 0.01347 seconds with return value 0
Press any key to continue . . .
```

```
- Warnings: 0
- Output Filename: C:\Users\Vattem Siva Sankar\Documents\System calls.exe
- Output Size: 127.9296875 KiB
- Compilation Time: 0.28s
```

C:\Users\Vattem Siva Sankar\Documents\System calls.c - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
program 7 fifo.cpp
program 9 optimal.cpp
program 8 lru.cpp
program 2.c
cpu scheduling 2.c
scheduling process small 2.c
bankers algorithm 2.c
[*] PROCEDURE CONSUMER .c
System calls.c
C:\Users\Vattem Siva Sankar\Documents\System calls.exe
Compiler
Resources
Abort Compilation
Shorten compiler paths
Line: 10 Col: 2 Sel: 0 Lines: 10 Length: 137 Insert Done parsing in 0 seconds
33°C Mostly cloudy
ENG
14:46
13-09-2022

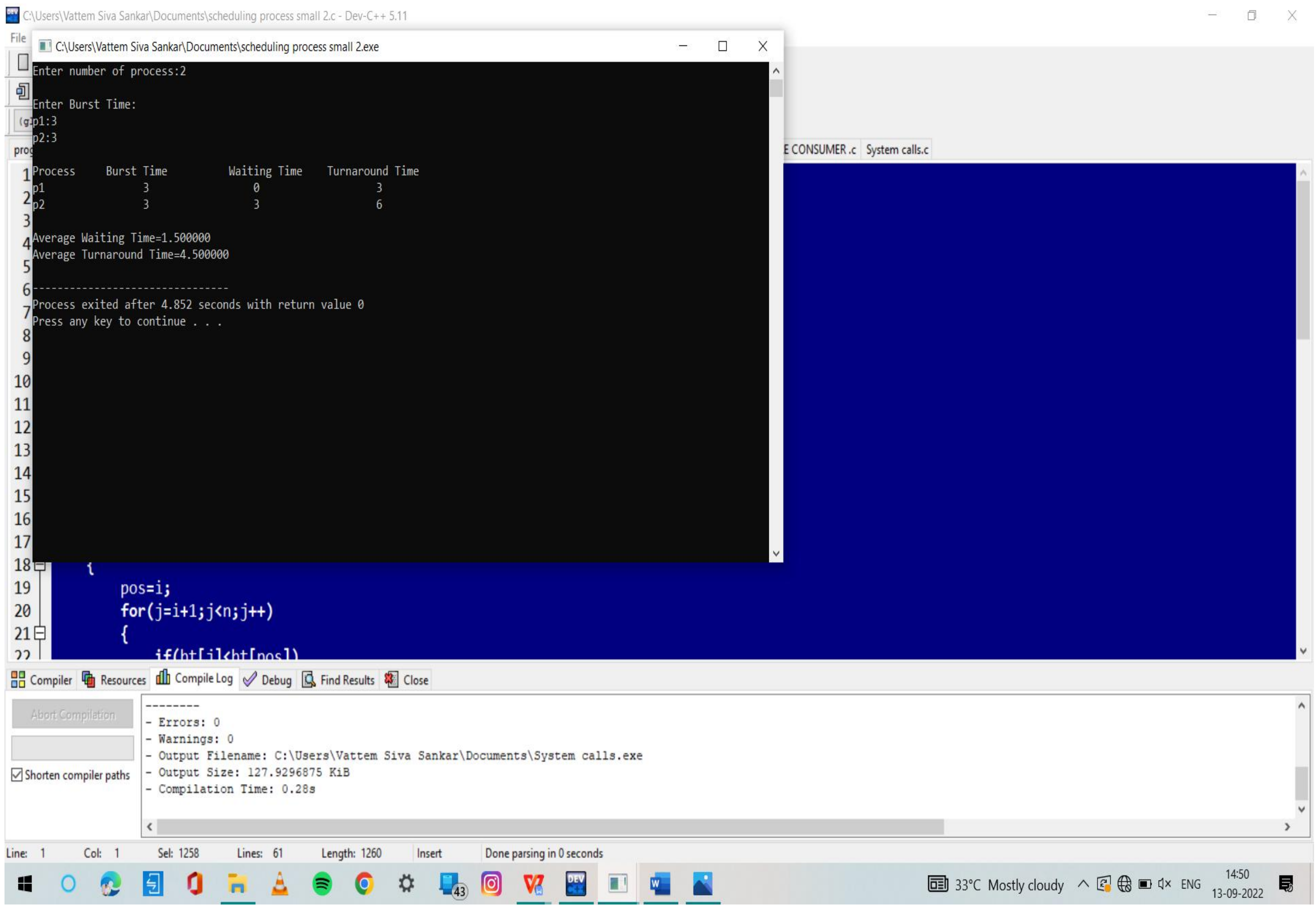
2.Identify the system calls to copy the content of one file to another and illustrate the same using a C program.

Program:

```
#include<stdio.h>


int main()

{

   int bt[20],p[20],wt[20],tat[20],i,j,n,total=0,pos,temp;

   float avg_wt,avg_tat;
```

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

printf("\nEnter Burst Time:\n");
for(i=0;i<n;i++)
{
    printf("p%d:",i+1);
    scanf("%d",&bt[i]);
    p[i]=i+1;
}
for(i=0;i<n;i++)
{
    pos=i;
    for(j=i+1;j<n;j++)
    {
        if(bt[j]<bt[pos])
            pos=j;
    }

    temp=bt[i];
    bt[i]=bt[pos];
    bt[pos]=temp;

    temp=p[i];
    p[i]=p[pos];
    p[pos]=temp;
}

wt[0]=0;

for(i=1;i<n;i++)
{
    wt[i]=0;
    for(j=0;j<i;j++)
        wt[i]+=bt[j];

    total+=wt[i];
}

avg_wt=(float)total/n;
total=0;

printf("\nProcess\t    Burst Time    \tWaiting Time\tTurnaround Time");
for(i=0;i<n;i++)
{
        tat[i]=bt[i]+wt[i];
        total+=tat[i];
        printf("\np%d\t\t  %d\t\t    %d\t\t\t%d",p[i],bt[i],wt[i],tat[i]);
    }

    avg_tat=(float)total/n;
    printf("\n\nAverage Waiting Time=%f",avg_wt);
    printf("\nAverage Turnaround Time=%f\n",avg_tat);
}
```

Output:


C:\Users\Vattem Siva Sankar\Documents\scheduling process small 2.c - Dev-C++ 5.11
C:\Users\Vattem Siva Sankar\Documents\scheduling process small 2.exe
E CONSUMER .c
System calls.c
Compiler
Resources
Compile Log
Debug
Find Results
Close
Abort Compilation
☑ Shorten compiler paths
Line: 1 Col: 1 Sel: 1258 Lines: 61 Length: 1260 Insert Done parsing in 0 seconds
33°C Mostly cloudy ENG 14:50 13-09-2022

```
Enter number of process:2

Enter Burst Time:
p1:3
p2:3

Process     Burst Time          Waiting Time    Turnaround Time
p1              3                   0                   3
p2              3                   3                   6

Average Waiting Time=1.500000
Average Turnaround Time=4.500000

--------------------------------
Process exited after 4.852 seconds with return value 0
Press any key to continue . . .
```

```
        {
            pos=i;
            for(j=i+1;j<n;j++)
            {
                if(bt[j]<bt[pos])
```

```
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\Vattem Siva Sankar\Documents\System calls.exe
- Output Size: 127.9296875 KiB
- Compilation Time: 0.28s
```

3. . DESIGN A CPU SCHEDULING PROGRAM WITH C USING FIRST COME FIRST SERVED TECHNIQUE WITH THE FOLLOWING CONSIDERATIONS.

A. ALL PROCESSES ARE ACTIVATED AT TIME 0.

B. ASSUME THAT NO PROCESS WAITS ON I/O DEVICES.

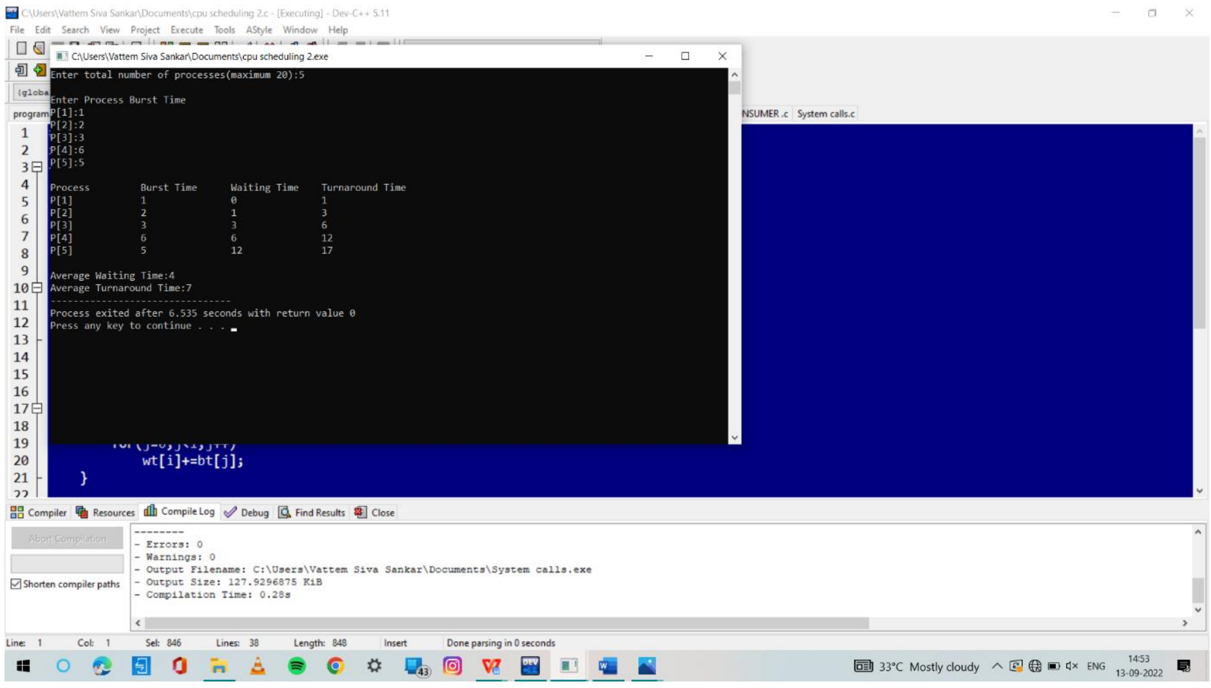
PROGRAM:

```
#include<stdio.h>

int main()

{

    int n,bt[20],wt[20],tat[20],avwt=0,avtat=0,i,j;
```

```
printf("Enter total number of processes(maximum 20):");
scanf("%d",&n);

printf("\nEnter Process Burst Time\n");
for(i=0;i<n;i++)
{
  printf("P[%d]:",i+1);
  scanf("%d",&bt[i]);
}

wt[0]=0;
for(i=1;i<n;i++)
{
  wt[i]=0;
  for(j=0;j<i;j++)
    wt[i]+=bt[j];
}

printf("\nProcess\t\tBurst Time\tWaiting Time\tTurnaround Time");
for(i=0;i<n;i++)
{
  tat[i]=bt[i]+wt[i];
  avwt+=wt[i];
  avtat+=tat[i];
  printf("\nP[%d]\t\t%d\t\t%d\t\t%d",i+1,bt[i],wt[i],tat[i]);
}

avwt/=i;
avtat/=i;
printf("\n\nAverage Waiting Time:%d",avwt);
printf("\nAverage Turnaround Time:%d",avtat);
```

```
    return 0;

}
```

Output:



C:\Users\Vattem Siva Sankar\Documents\cpu scheduling 2.c - [Executing] - Dev-C++ 5.11

C:\Users\Vattem Siva Sankar\Documents\cpu scheduling 2.exe

```
Enter total number of processes(maximum 20):5

Enter Process Burst Time
P[1]:1
P[2]:2
P[3]:3
P[4]:6
P[5]:5

Process         Burst Time      Waiting Time    Turnaround Time
P[1]            1               0               1
P[2]            2               1               3
P[3]            3               3               6
P[4]            6               6               12
P[5]            5               12              17

Average Waiting Time:4
Average Turnaround Time:7
--------------------------------
Process exited after 6.535 seconds with return value 0
Press any key to continue . . .
```

```
            wt[i]+=bt[j];
    }
```

```
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\Vattem Siva Sankar\Documents\System calls.exe
- Output Size: 127.9296875 KiB
- Compilation Time: 0.28s
```

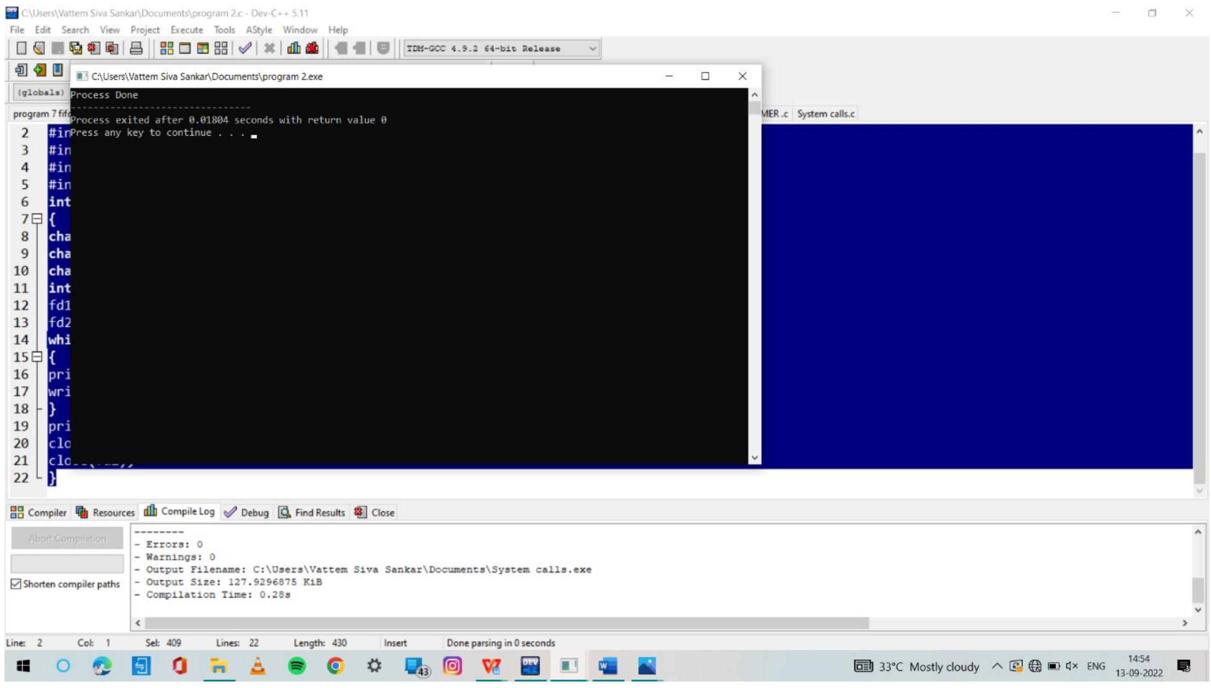
4. . Construct a scheduling program with C that selects the waiting process with the smallest execution time to execute next.

PROGRAM:

```
. #include<sys/types.h>

#include<fcntl.h>

#include<stdlib.h>

#include<string.h>

int main(int args,char *ar[])

{

char *source=ar[1];

char *dest="def.txt";

char *buf=(char *)malloc(sizeof(char)*120);

int fd1,fd2;

fd1=open(source,O_CREAT,0744);

fd2=open(dest,O_CREAT,0744);

while(read(fd1,buf,120)!=-1)

{
```

```
printf("%s",buf);

write(fd2,buf,120);

}

printf("Process Done");

close(fd1);

close(fd2);

}
```

Output:


C:\Users\Vattem Siva Sankar\Documents\program 2.c - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
C:\Users\Vattem Siva Sankar\Documents\program 2.exe
(globals)
MER .c
System calls.c
Compiler
Resources
Compile Log
Debug
Find Results
Close
Abort Compilation
☑ Shorten compiler paths
Line: 2 Col: 1 Sel: 409 Lines: 22 Length: 430 Insert Done parsing in 0 seconds
33°C Mostly cloudy
ENG
14:54
13-09-2022

```
Process Done
--------------------------------
Process exited after 0.01804 seconds with return value 0
Press any key to continue . . .
```

```
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\Vattem Siva Sankar\Documents\System calls.exe
- Output Size: 127.9296875 KiB
- Compilation Time: 0.28s
```

5. Illustrate the deadlock avoidance concept by simulating Banker’s algorithm with C.

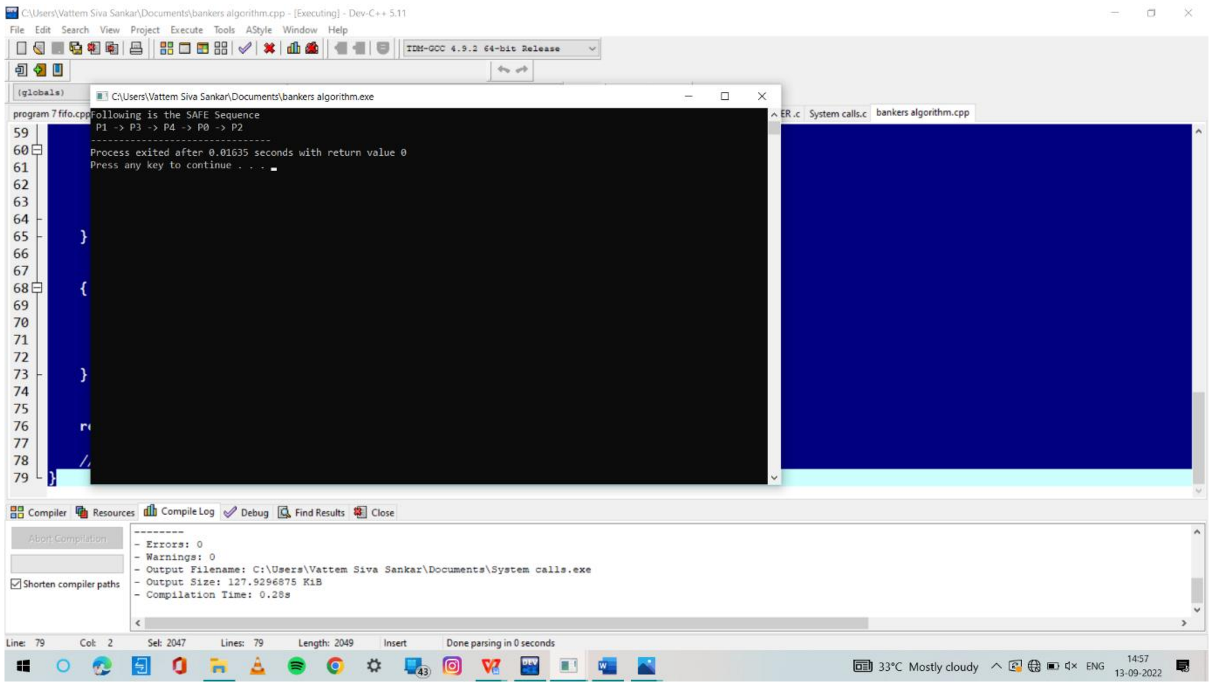
Program:

```
#include <stdio.h>

int main()

{

   // P0, P1, P2, P3, P4 are the Process names here


   int n, m, i, j, k;

   n = 5; // Number of processes

   m = 3; // Number of resources

   int alloc[5][3] = { { 0, 1, 0 }, // P0    // Allocation Matrix

                       { 2, 0, 0 }, // P1

                       { 3, 0, 2 }, // P2

                       { 2, 1, 1 }, // P3

                       { 0, 0, 2 } }; // P4


   int max[5][3] = { { 7, 5, 3 }, // P0    // MAX Matrix

                     { 3, 2, 2 }, // P1

                     { 9, 0, 2 }, // P2

                     { 2, 2, 2 }, // P3
```

```
                { 4, 3, 3 } }; // P4

int avail[3] = { 3, 3, 2 }; // Available Resources

int f[n], ans[n], ind = 0;
for (k = 0; k < n; k++) {
    f[k] = 0;
}
int need[n][m];
for (i = 0; i < n; i++) {
    for (j = 0; j < m; j++)
        need[i][j] = max[i][j] - alloc[i][j];
}
int y = 0;
for (k = 0; k < 5; k++) {
    for (i = 0; i < n; i++) {
        if (f[i] == 0) {

            int flag = 0;
            for (j = 0; j < m; j++) {
                if (need[i][j] > avail[j]){
                    flag = 1;
                     break;
                }
            }

            if (flag == 0) {
                ans[ind++] = i;
                for (y = 0; y < m; y++)
                    avail[y] += alloc[i][y];
                f[i] = 1;
            }
        }
    }
}

  int flag = 1;

  for(int i=0;i<n;i++)
{
  if(f[i]==0)
  {
    flag=0;
     printf("The following system is not safe");
    break;
```

```
        }
      }


      if(flag==1)
    {
      printf("Following is the SAFE Sequence\n");
      for (i = 0; i < n - 1; i++)
        printf(" P%d ->", ans[i]);
      printf(" P%d", ans[n - 1]);
    }



    return (0);


}
```

Output:



C:\Users\Vattem Siva Sankar\Documents\bankers algorithm.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
C:\Users\Vattem Siva Sankar\Documents\bankers algorithm.exe
program 7 fifo.cpp
ER.c
System calls.c
bankers algorithm.cpp
Compiler
Resources
Compile Log
Debug
Find Results
Close
Abort Compilation
☑ Shorten compiler paths
Line: 79
Col: 2
Sel: 2047
Lines: 79
Length: 2049
Insert
Done parsing in 0 seconds
33°C Mostly cloudy
ENG
14:57
13-09-2022

```
Following is the SAFE Sequence
 P1 -> P3 -> P4 -> P0 -> P2
--------------------------------
Process exited after 0.01635 seconds with return value 0
Press any key to continue . . .
```

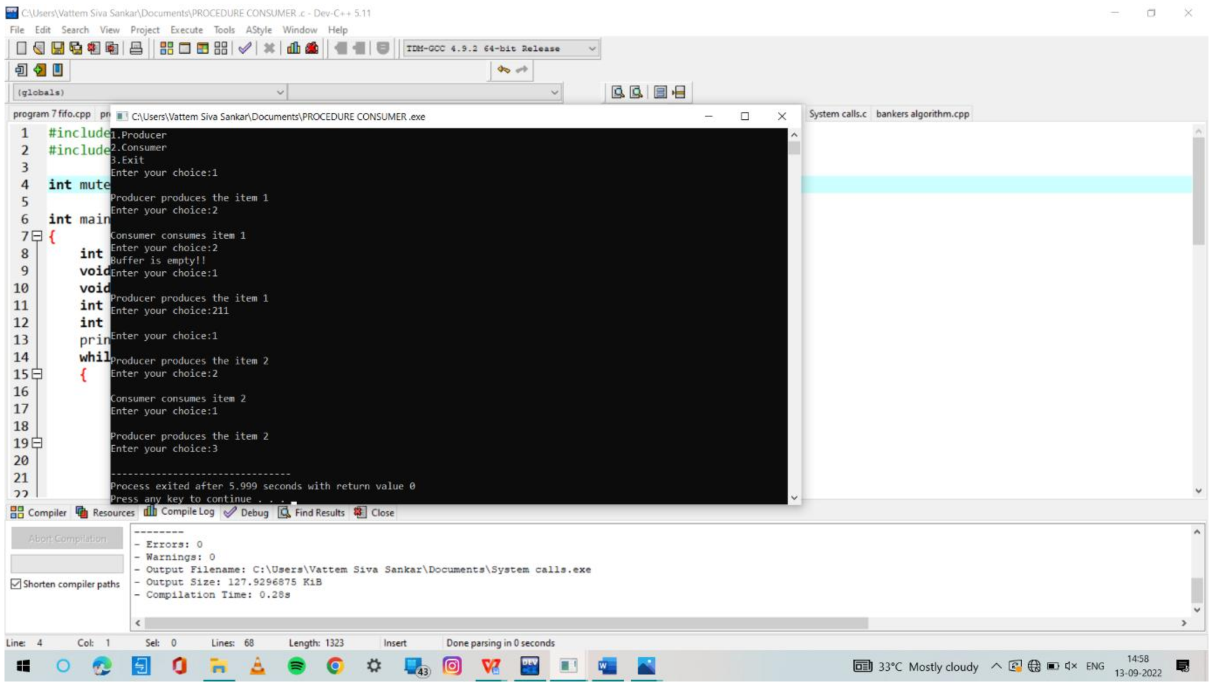
```
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\Vattem Siva Sankar\Documents\System calls.exe
- Output Size: 127.9296875 KiB
- Compilation Time: 0.28s
```

6. Construct a C program to simulate producer-consumer problem using semaphores.

Program:

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

int mutex=1,full=0,empty=3,x=0;
int main()
{
    int n;
    void producer();
    void consumer();
    int wait(int);
    int signal(int);
    printf("\n1.Producer\n2.Consumer\n3.Exit");
    while(1)
    {
        printf("\nEnter your choice:");
        scanf("%d",&n);
        switch(n)
        {
            case 1:    if((mutex==1)&&(empty!=0))
                        producer();
                    else
                        printf("Buffer is full!!");
                    break;
            case 2:    if((mutex==1)&&(full!=0))
                        consumer();
                    else
                        printf("Buffer is empty!!");
                    break;
            case 3:
                    exit(0);
                    break;
        }
    }

    return 0;
}

int wait(int s)
{
    return (--s);
}

int signal(int s)
{
    return(++s);
}

void producer()
{
    mutex=wait(mutex);

    full=signal(full);

    empty=wait(empty);

    x++;

    printf("\nProducer produces the item %d",x);

    mutex=signal(mutex);

}


void consumer()

{

    mutex=wait(mutex);

    full=wait(full);

    empty=signal(empty);

    printf("\nConsumer consumes item %d",x);

    x--;

    mutex=signal(mutex);

}
```

Output:



```
1.Producer
2.Consumer
3.Exit
Enter your choice:1

Producer produces the item 1
Enter your choice:2

Consumer consumes item 1
Enter your choice:2
Buffer is empty!!
Enter your choice:1

Producer produces the item 1
Enter your choice:211

Enter your choice:1

Producer produces the item 2
Enter your choice:2

Consumer consumes item 2
Enter your choice:1

Producer produces the item 2
Enter your choice:3

--------------------------------
Process exited after 5.999 seconds with return value 0
Press any key to continue . . .
```

```
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\Vattem Siva Sankar\Documents\System calls.exe
- Output Size: 127.9296875 KiB
- Compilation Time: 0.28s
```

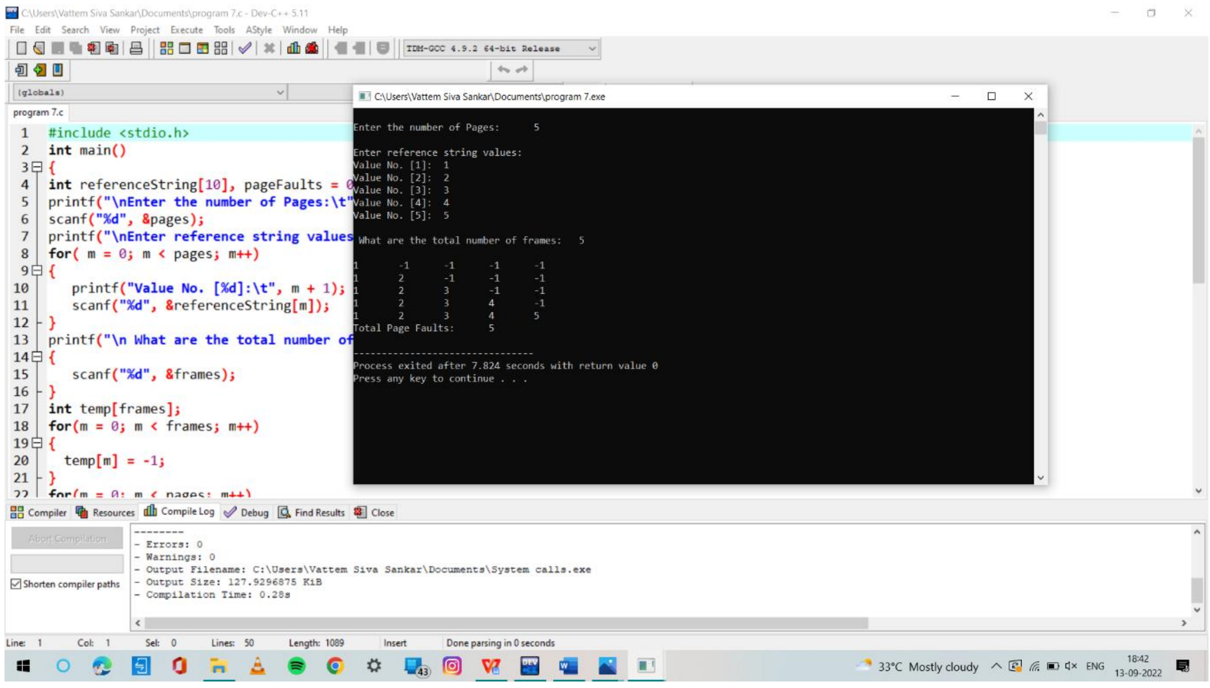
7. . Construct a C program to simulate the First in First Out paging technique of memory management.

Program:

```
#include <stdio.h>

int main()

{

int referenceString[10], pageFaults = 0, m, n, s, pages, frames;
```

```
printf("\nEnter the number of Pages:\t");

scanf("%d", &pages);

printf("\nEnter reference string values:\n");

for( m = 0; m < pages; m++)

{

   printf("Value No. [%d]:\t", m + 1);

   scanf("%d", &referenceString[m]);

}

printf("\n What are the total number of frames:\t");

{

   scanf("%d", &frames);

}

int temp[frames];

for(m = 0; m < frames; m++)

{

  temp[m] = -1;

}

for(m = 0; m < pages; m++)

{

  s = 0;

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

    {

      if(referenceString[m] == temp[n])

         {

            s++;

            pageFaults--;

         }

    }

   pageFaults++;

   if((pageFaults <= frames) && (s == 0))

      {
        temp[m] = referenceString[m];

      }

   else if(s == 0)

      {

        temp[(pageFaults - 1) % frames] = referenceString[m];

      }

      printf("\n");

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

       {

         printf("%d\t", temp[n]);

       }

}

printf("\nTotal Page Faults:\t%d\n", pageFaults);

return 0;

}
```

Output:



C:\Users\Vattem Siva Sankar\Documents\program 7.c - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
program 7.c
C:\Users\Vattem Siva Sankar\Documents\program 7.exe
Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
☑ Shorten compiler paths
Line: 1 Col: 1 Sel: 0 Lines: 50 Length: 1089 Insert Done parsing in 0 seconds
33°C Mostly cloudy ENG 18:42 13-09-2022

```
#include <stdio.h>
int main()
{
int referenceString[10], pageFaults = 0
printf("\nEnter the number of Pages:\t"
scanf("%d", &pages);
printf("\nEnter reference string values
for( m = 0; m < pages; m++)
{
   printf("Value No. [%d]:\t", m + 1);
   scanf("%d", &referenceString[m]);
}
printf("\n What are the total number of
{
   scanf("%d", &frames);
}
int temp[frames];
for(m = 0; m < frames; m++)
{
  temp[m] = -1;
}
for(m = 0; m < pages; m++)
```

```
Enter the number of Pages:      5

Enter reference string values:
Value No. [1]:  1
Value No. [2]:  2
Value No. [3]:  3
Value No. [4]:  4
Value No. [5]:  5

 What are the total number of frames:   5

1       -1      -1      -1      -1
1       2       -1      -1      -1
1       2       3       -1      -1
1       2       3       4       -1
1       2       3       4       5
Total Page Faults:      5

--------------------------------
Process exited after 7.824 seconds with return value 0
Press any key to continue . . .
```

```
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\Vattem Siva Sankar\Documents\System calls.exe
- Output Size: 127.9296875 KiB
- Compilation Time: 0.28s
```

8. . Construct a C program to simulate the Least Recently Used paging technique of memory management.

Program:

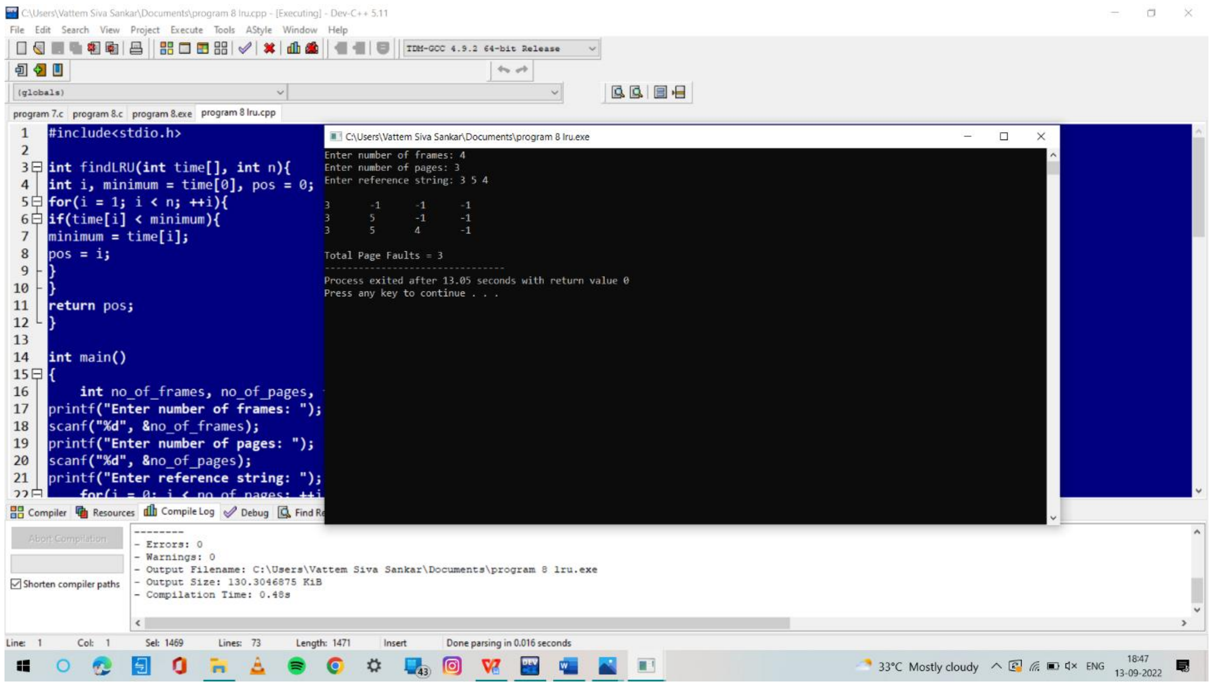
```
#include<stdio.h>
```

```
int findLRU(int time[], int n){

int i, minimum = time[0], pos = 0;

for(i = 1; i < n; ++i){

if(time[i] < minimum){

minimum = time[i];

pos = i;

}

}

return pos;

}



int main()

{

    int no_of_frames, no_of_pages, frames[10], pages[30], counter = 0, time[10], flag1, flag2, i, j, pos, faults = 0;

printf("Enter number of frames: ");

scanf("%d", &no_of_frames);

printf("Enter number of pages: ");

scanf("%d", &no_of_pages);

printf("Enter reference string: ");

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


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

    }


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

     frames[i] = -1;

    }


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

     flag1 = flag2 = 0;
```

```
   for(j = 0; j < no_of_frames; ++j){

   if(frames[j] == pages[i]){

   counter++;

   time[j] = counter;

  flag1 = flag2 = 1;

  break;

  }

   }



   if(flag1 == 0){

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

   if(frames[j] == -1){

   counter++;

   faults++;

   frames[j] = pages[i];

   time[j] = counter;

   flag2 = 1;

   break;

   }

   }

   }



   if(flag2 == 0){

   pos = findLRU(time, no_of_frames);

   counter++;

   faults++;

   frames[pos] = pages[i];

   time[pos] = counter;

   }
```

```
    printf("\n");


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

    printf("%d\t", frames[j]);

    }

}

printf("\n\nTotal Page Faults = %d", faults);



   return 0;

}
```

Output:



C:\Users\Vattem Siva Sankar\Documents\program 8 lru.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
program 7.c
program 8.c
program 8.exe
program 8 lru.cpp
C:\Users\Vattem Siva Sankar\Documents\program 8 lru.exe
Compiler
Resources
Compile Log
Debug
Find Re
Abort Compilation
Shorten compiler paths
Line: 1 Col: 1 Sel: 1469 Lines: 73 Length: 1471 Insert Done parsing in 0.016 seconds
33°C Mostly cloudy
ENG
18:47
13-09-2022

```
#include<stdio.h>

int findLRU(int time[], int n){
int i, minimum = time[0], pos = 0;
for(i = 1; i < n; ++i){
if(time[i] < minimum){
minimum = time[i];
pos = i;
}
}
return pos;
}

int main()
{
    int no_of_frames, no_of_pages,
printf("Enter number of frames: ");
scanf("%d", &no_of_frames);
printf("Enter number of pages: ");
scanf("%d", &no_of_pages);
printf("Enter reference string: ");
    for(i = 0; i < no_of_pages; ++i
```

```
Enter number of frames: 4
Enter number of pages: 3
Enter reference string: 3 5 4

3       -1      -1      -1
3       5       -1      -1
3       5       4       -1

Total Page Faults = 3
--------------------------------
Process exited after 13.05 seconds with return value 0
Press any key to continue . . .
```

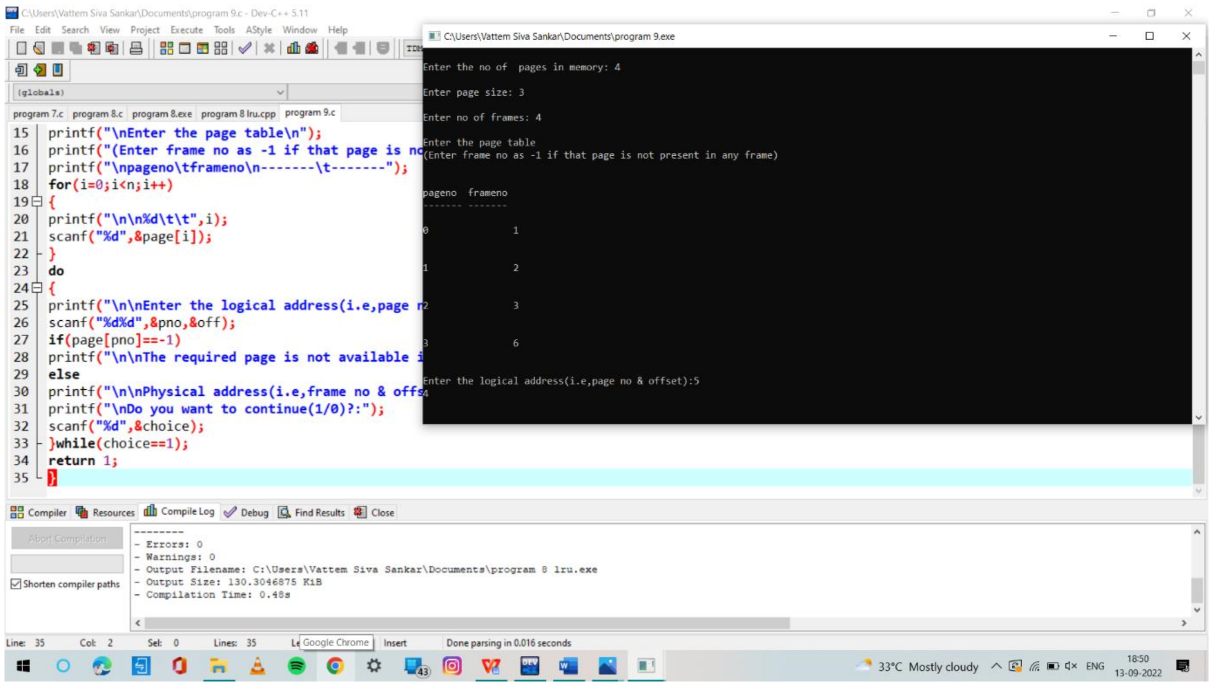
```
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\Vattem Siva Sankar\Documents\program 8 lru.exe
- Output Size: 130.3046875 KiB
- Compilation Time: 0.48s
```

9. Construct a C program to simulate the optimal paging technique of memory management.

Program:

```
#include<stdio.h>

#define MAX 50

int main()

{

int page[MAX],i,n,f,ps,off,pno;

int choice=0;
printf("\nEnter the no of  pages in memory: ");
scanf("%d",&n);
printf("\nEnter page size: ");
scanf("%d",&ps);
printf("\nEnter no of frames: ");
scanf("%d",&f);
for(i=0;i<n;i++)
page[i]=-1;
printf("\nEnter the page table\n");
printf("(Enter frame no as -1 if that page is not present in any frame)\n\n");
printf("\npageno\tframeno\n-------\t-------");
for(i=0;i<n;i++)
{
printf("\n\n%d\t\t",i);
scanf("%d",&page[i]);
}
do
{
printf("\n\nEnter the logical address(i.e,page no & offset):");
scanf("%d%d",&pno,&off);
if(page[pno]==-1)
printf("\n\nThe required page is not available in any of frames");
else
printf("\n\nPhysical address(i.e,frame no & offset):%d,%d",page[pno],off);
printf("\nDo you want to continue(1/0)?:");
scanf("%d",&choice);
}while(choice==1);
return 1;
}
```

Output:



C:\Users\Vattem Siva Sankar\Documents\program 9.c - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
(globals)
program 7.c
program 8.c
program 8.exe
program 8 lru.cpp
program 9.c

```c
printf("\nEnter the page table\n");
printf("(Enter frame no as -1 if that page is no
printf("\npageno\tframeno\n-------\t-------");
for(i=0;i<n;i++)
{
printf("\n\n%d\t\t",i);
scanf("%d",&page[i]);
}
do
{
printf("\n\nEnter the logical address(i.e,page n
scanf("%d%d",&pno,&off);
if(page[pno]==-1)
printf("\n\nThe required page is not available i
else
printf("\n\nPhysical address(i.e,frame no & offs
printf("\nDo you want to continue(1/0)?:");
scanf("%d",&choice);
}while(choice==1);
return 1;
}
```

```
Enter the no of  pages in memory: 4

Enter page size: 3

Enter no of frames: 4

Enter the page table
(Enter frame no as -1 if that page is not present in any frame)


pageno  frameno
------- -------

0              1


1              2


2              3


3              6


Enter the logical address(i.e,page no & offset):5
4
```

```
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\Vattem Siva Sankar\Documents\program 8 lru.exe
- Output Size: 130.3046875 KiB
- Compilation Time: 0.48s
```