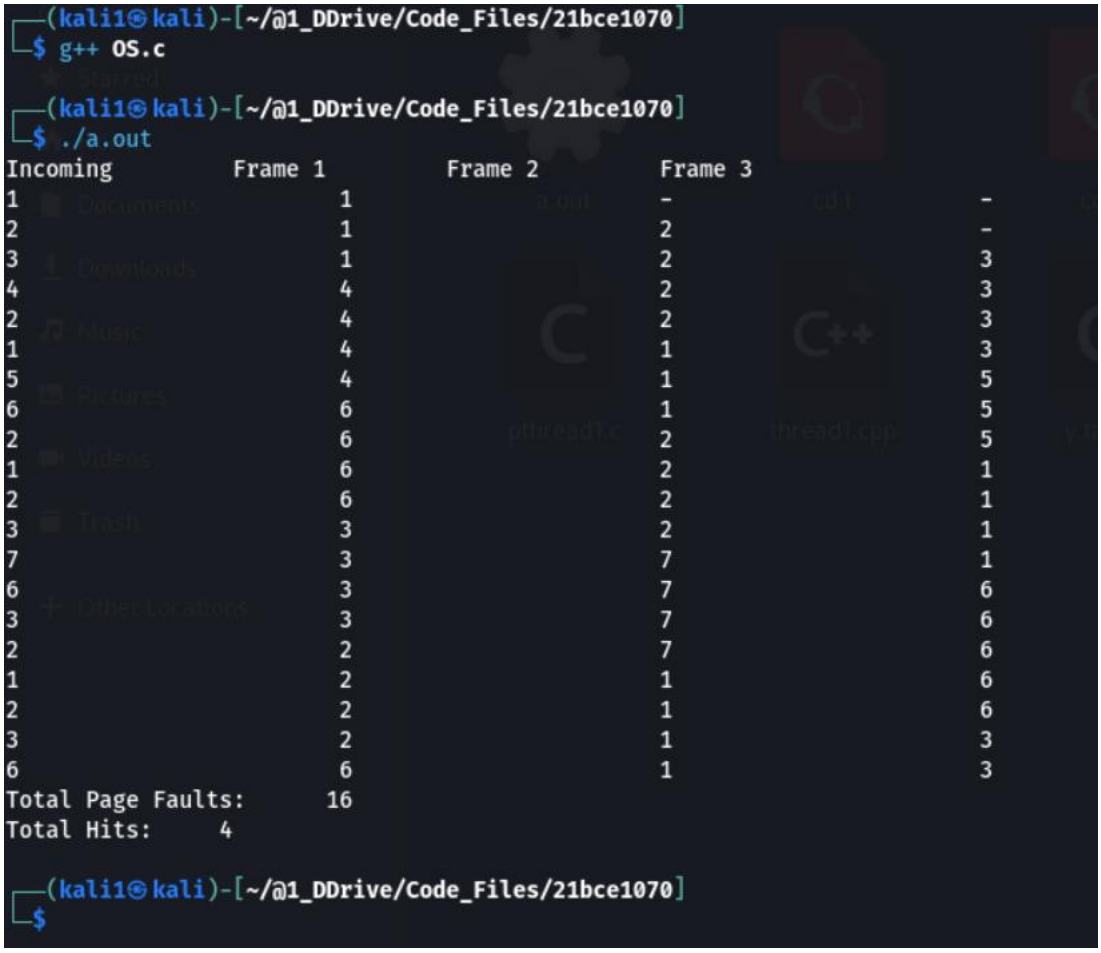
Page Replacement

First In First Out

```
#include<stdio.h>
int main()
{
   int incomingStream[] = {1,2,2,4,2,1,5,6,2,1,2,3,7,6,3,2,1,2,3,6};
   int pageFaults = 0;
   int frames = 3;
   int m, n, s, pages;

   pages = sizeof(incomingStream)/sizeof(incomingStream[0]);

   printf("Incoming \t Frame 1 \t Frame 2 \t Frame 3");
   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(incomingStream[m] == temp[n])
         {
            s++;
            pageFaults--;
```

```
            }
        }
        pageFaults++;

        if((pageFaults <= frames) && (s == 0))
        {
            temp[pageFaults-1] = incomingStream[m];
        }
        else if(s == 0)
        {
            temp[(pageFaults - 1) % frames] = incomingStream[m];
        }

        printf("\n");
        printf("%d\t\t\t",incomingStream[m]);
        for(n = 0; n < frames; n++)
        {
            if(temp[n] != -1)
                printf(" %d\t\t\t", temp[n]);
            else
                printf(" - \t\t\t");
        }
    }

printf("\nTotal Page Faults:\t%d\nTotal Hits:\t%d\n", pageFaults,(pages-pageFaults));
    return 0;
}
```

Output



Least Recently Used

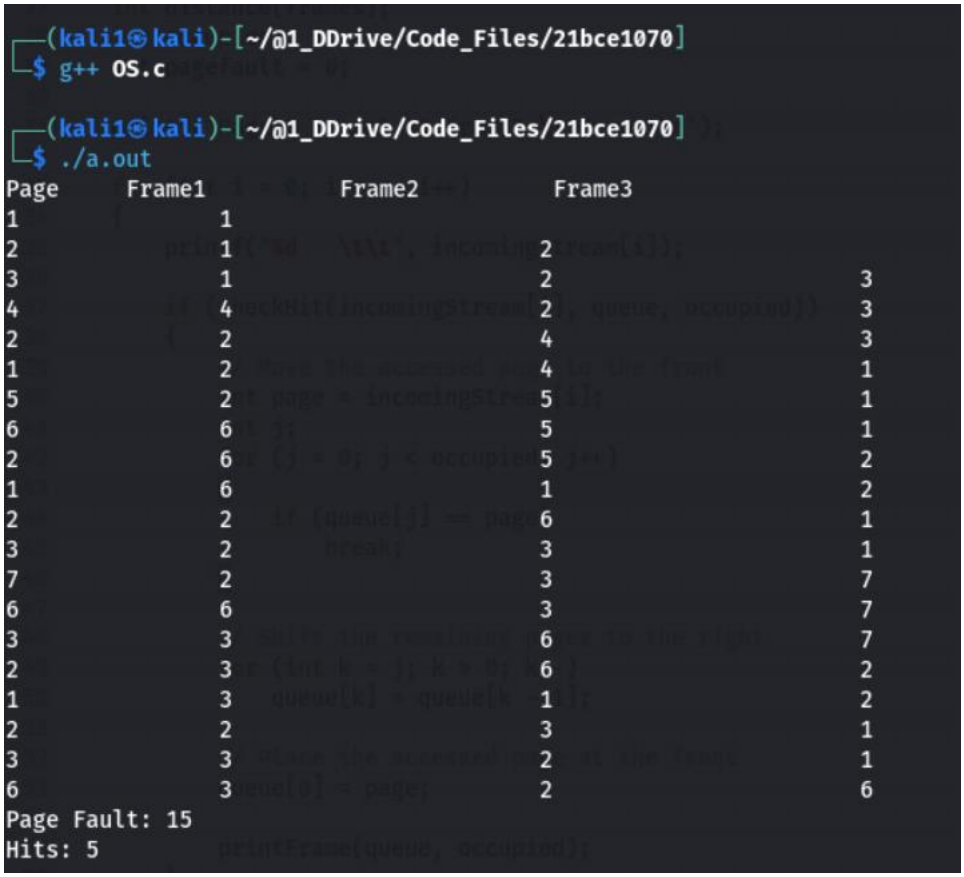
```
#include <stdio.h>
#include <limits.h>

int checkHit(int incomingPage, int queue[], int occupied)
{
   for (int i = 0; i < occupied; i++)
   {
      if (incomingPage == queue[i])
         return 1;
   }
```

```
    return 0;
}

void printFrame(int queue[], int occupied)
{
    for (int i = 0; i < occupied; i++)
        printf("%d\t\t\t", queue[i]);
}

int main()
{
    int incomingStream[] = {1,2,3,4,2,1,5,6,2,1,2,3,7,6,3,2,1,2,3,6};
    int n = sizeof(incomingStream) / sizeof(incomingStream[0]);
    int frames = 3;
    int queue[frames];
    int distance[frames];
    int occupied = 0;
    int pagefault = 0;

    printf("Page\t Frame1 \t Frame2 \t Frame3\n");

    for (int i = 0; i < n; i++)
    {
        printf("%d   \t\t", incomingStream[i]);

        if (checkHit(incomingStream[i], queue, occupied))
        {
            // Move the accessed page to the front
            int page = incomingStream[i];
            int j;
            for (j = 0; j < occupied; j++)
```

```
    {
        if (queue[j] == page)
            break;
    }

    // Shift the remaining pages to the right
    for (int k = j; k > 0; k--)
        queue[k] = queue[k - 1];

    // Place the accessed page at the front
    queue[0] = page;

    printFrame(queue, occupied);
}
else if (occupied < frames)
{
    queue[occupied] = incomingStream[i];
    occupied++;

    printFrame(queue, occupied);
    pagefault++;
}
else
{
    int max = INT_MIN;
    int index;

    for (int j = 0; j < frames; j++)
    {
        distance[j] = 0;
```

```
            for (int k = i - 1; k >= 0; k--)
            {
               ++distance[j];

               if (queue[j] == incomingStream[k])
                  break;
            }

            if (distance[j] > max)
            {
               max = distance[j];
               index = j;
            }
         }

         queue[index] = incomingStream[i];
         printFrame(queue, occupied);
         pagefault++;
      }

      printf("\n");
   }

   printf("Page Fault: %d\nHits: %d\n", pagefault,n-pagefault);

   return 0;
}
```

Output



Optimized

```
#include <stdio.h>


int search(int key, int frame_items[], int frame_occupied)
{
   for (int i = 0; i < frame_occupied; i++)
      if (frame_items[i] == key)
         return 1;
   return 0;
}


void printOuterStructure(int max_frames)
{
   printf("Stream ");
```

```
    for (int i = 0; i < max_frames; i++)
        printf("Frame%d ", i + 1);
}

void printCurrFrames(int item, int frame_items[], int frame_occupied, int max_frames)
{
    printf("\n%d \t\t", item);
    for (int i = 0; i < max_frames; i++)
    {
        if (i < frame_occupied)
            printf("%d \t\t", frame_items[i]);
        else
            printf("- \t\t");
    }
}

int predict(int ref_str[], int frame_items[], int refStrLen, int index, int frame_occupied)
{
    int result = -1, farthest = index;
    for (int i = 0; i < frame_occupied; i++)
    {
        int j;
        for (j = index; j < refStrLen; j++)
        {
            if (frame_items[i] == ref_str[j])
            {
                if (j > farthest)
                {
                    farthest = j;
                    result = i;
                }
```

```
            break;
        }
    }
    if (j == refStrLen)
        return i;
  }
  return (result == -1) ? 0 : result;
}

void optimalPage(int ref_str[], int refStrLen, int frame_items[], int max_frames)
{
  int frame_occupied = 0;
  printOuterStructure(max_frames);
  int hits = 0;
  for (int i = 0; i < refStrLen; i++)
  {
    if (search(ref_str[i], frame_items, frame_occupied))
    {
      hits++;
      printCurrFrames(ref_str[i], frame_items, frame_occupied, max_frames);
      continue;
    }
    if (frame_occupied < max_frames)
    {
      frame_items[frame_occupied] = ref_str[i];
      frame_occupied++;
      printCurrFrames(ref_str[i], frame_items, frame_occupied, max_frames);
    }
    else
    {
      int pos = predict(ref_str, frame_items, refStrLen, i + 1, frame_occupied);
```

```
            frame_items[pos] = ref_str[i];
            printCurrFrames(ref_str[i], frame_items, frame_occupied, max_frames);
        }
    }
    printf("\n\nHits: %d\n", hits);
    printf("Misses: %d", refStrLen - hits);
}

int main()
{
    int ref_str[] = {1,2,3,4,2,1,5,6,2,1,2,3,7,6,3,2,1,2,3,6};
    int refStrLen = sizeof(ref_str) / sizeof(ref_str[0]);
    int max_frames = 3;
    int frame_items[max_frames];

    optimalPage(ref_str, refStrLen, frame_items, max_frames);
    return 0;
}
```

Output

```
┌──(kali1㉿kali)-[~/@1_DDrive/Code_Files/21bce1070]
└─$ g++ OS.c

┌──(kali1㉿kali)-[~/@1_DDrive/Code_Files/21bce1070]
└─$ ./a.out
Stream Frame1 Frame2 Frame3
1              1              -              -
2              1              2              -
3              1              2              3
4              1              2              4
2              1              2              4
1              1              2              4
5              1              2              5
6              1              2              6
2              1              2              6
1              1              2              6
2              1              2              6
3              3              2              6
7              3              7              6
6              3              7              6
3              3              7              6
2              3              2              6
1              3              2              1
2              3              2              1
3              3              2              1
6              6              2              1

Hits: 9
Misses: 11
```