Name : Ashutosh Ardu

Regno : 20BRS1262

# OS LAB FAT

1. Class representatives A,B,C,D are going to our library to lend some books to meet their requirements. They want to get DS, OS, CN and OOPS from library for their class students. The total amount of books for each subject in library is (16, 5, 2, 8). The maximum requirements of books in each subject for each of them is given as A(4,4,2,1), B(4,3,1,1), C(13,5,2,7) and D(6,1,1,1). The books already taken or issued to the students in the class are given as A(4,0,0,1),B(1,2,1,0),C(1,1,0,2) and D(3,1,1,0). Assume books will be given back by class representative so that librarian can satisfy the requirements. Illustrate which allotment sequence will satisfy all the requirements of the class representative.

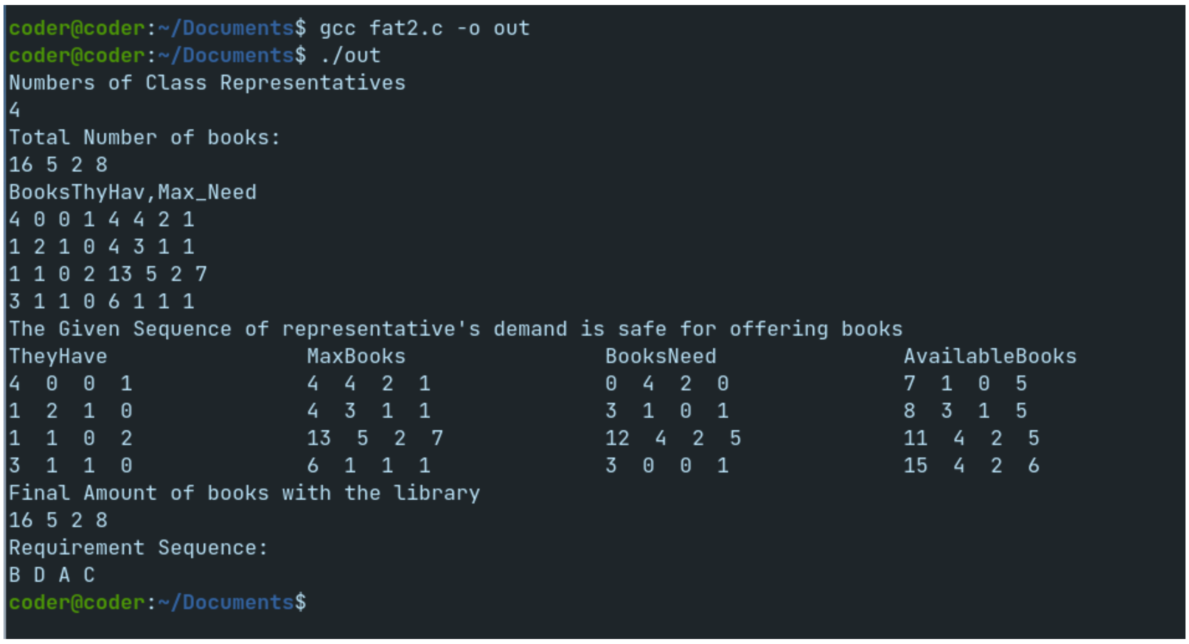
## Code

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

int order[100];
int k=0;

struct job{
    int a,b,c,d,finish,n;
}avail;
int comp(struct job a,struct job *b){
    if(a.a<=b->a && a.b<=b->b && a.c<=b->c && a.d<=b->d)
return 1;
    return 0;
}

void sum(struct job *r,struct job *proc){
        r->a+=proc->a;
        r->b+=proc->b;
        r->c+=proc->c;
        r->d+=proc->d;
    }

    void get(struct job *proc,int a,int i){
        scanf("%d %d %d %d",&proc->a,&proc->b,&proc->c,&proc->d);
        proc->finish=0;
        proc->n=i+1;
        if(a==1){
            sum(&avail,proc);
        }
    }

    void assign(struct job a,struct job proc[],int i){
        proc[i].a=a.a;
        proc[i].b=a.b;
        proc[i].c=a.c;
        proc[i].d=a.d;
    }

    void put(struct job *x,struct job *y,struct job *z){
        x->a=z->a-y->a;
        x->b=z->b-y->b;
        x->c=z->c-y->c;
        x->d=z->d-y->d;
        x->n=z->n;
        x->finish=0;
    }

    void append(struct job *a,struct job b){
        a->a+=b.a;
        a->b+=b.b;
        a->c+=b.c;
        a->d+=b.d;
    }

    int check(struct job remain[],struct job *a,struct job
    b[],struct job current[],int *t,int n){
        for(int i=0;i<n;++i){
            if(!remain[i].finish){
```

```c
            if(comp(remain[i],a)){
                remain[i].finish=1;
                order[k++]=remain[i].n;
                append(a,b[i]);
                assign(avail,current,*t);*t+=1;
                return 1;
            }
        }
    }return 0;
}

void print(struct job a[],struct job b[],struct job c[],struct
job d[],int n){
    printf("TheyHave\t\tMaxBooks\t\tBooksNeed\t\tAvailableBook
s\n");
    for(int i=0;i<n;++i){
    printf("%d  %d  %d  %d\t\t",a[i].a,a[i].b,a[i].c,a[i].d);
    printf("%d  %d  %d  %d\t\t",b[i].a,b[i].b,b[i].c,b[i].d);
    printf("%d  %d  %d  %d\t\t",c[i].a,c[i].b,c[i].c,c[i].d);
    printf("%d  %d  %d  %d\n",d[i].a,d[i].b,d[i].c,d[i].d);
    }
}

void copy(struct job a[],struct job b[],int i){
    a[i].a=b[i].a;
    a[i].b=b[i].b;
    a[i].c=b[i].c;
    a[i].d=b[i].d;
}

int main(){
    printf("Numbers of Class Representatives\n");
    int n,s=0,t=0;scanf("%d",&n);
    struct job
need[n],total,remain[n],alloc[n],current[n],alloc1[n];
    avail.a=avail.b=avail.c=avail.d=0;
    printf("Total Number of books:\n");get(&total,0,0);
    printf("BooksThyHav,Max_Need\n");
    for(int i=0;i<n;++i){
        get(&alloc[i],1,i);
        copy(alloc1,alloc,i);
        get(&need[i],0,i);
```

```
            put(&remain[i],&alloc[i],&need[i]);
        }avail.a=total.a-avail.a;
        avail.b=total.b-avail.b;
        avail.c=total.c-avail.c;
        avail.d=total.d-avail.d;
        assign(avail,current,t);t++;
        for(int i=0;i<n;++i){
            if(!(check(remain,&avail,alloc,current,&t,n))){
                printf("Deadlock Occured\nAll Class reps will 
never be able to get what they want\n");
                break;s=1;
            }
        }
        if(s==0){
            printf("The Given Sequence of representative's demand 
is safe for offering books\n");
            print(alloc1,need,remain,current,n);
            printf("Final Amount of books with the library\n");
            printf("%d %d %d 
%d\n",avail.a,avail.b,avail.c,avail.d);
            printf("Requirement Sequence:\n");
            for(int i=0;i<k;++i)
                printf("%c ",(char)order[i]+64);
            printf("\n");
        }
    }
```

## Output



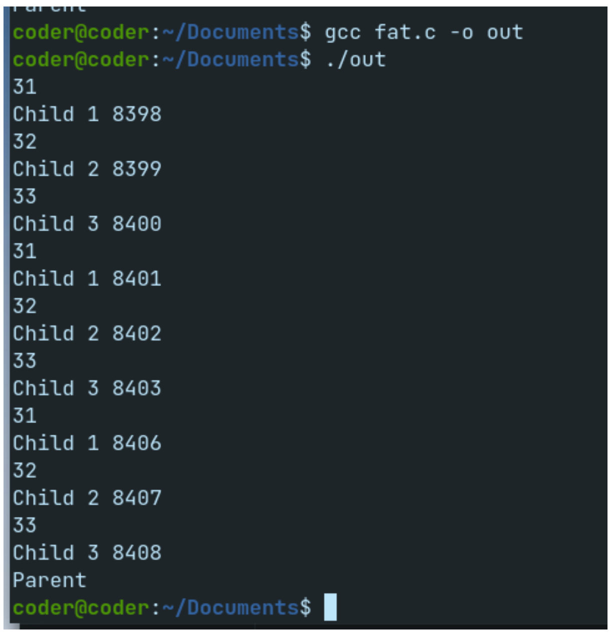
```
coder@coder:~/Documents$ gcc fat2.c -o out
coder@coder:~/Documents$ ./out
Numbers of Class Representatives
4
Total Number of books:
16 5 2 8
BooksThyHav,Max_Need
4 0 0 1 4 4 2 1
1 2 1 0 4 3 1 1
1 1 0 2 13 5 2 7
3 1 1 0 6 1 1 1
The Given Sequence of representative's demand is safe for offering books
TheyHave                MaxBooks                BooksNeed               AvailableBooks
4  0  0  1              4  4  2  1              0  4  2  0              7  1  0  5
1  2  1  0              4  3  1  1              3  1  0  1              8  3  1  5
1  1  0  2              13  5  2  7             12  4  2  5             11  4  2  5
3  1  1  0              6  1  1  1              3  0  0  1              15  4  2  6
Final Amount of books with the library
16 5 2 8
Requirement Sequence:
B D A C
coder@coder:~/Documents$
```

2. Write a C code to create a parent with three children. Each child should perform a unique arithmetic operation and print the result along with its process id. Only one child process can execute at a time and other child process should be blocked. The process that completes a task will wake-up the next process to execute. Starting from child 1 to 3 implement the above step for ‘n’ iterations. After ’n’ iteration, child 3 should start executing and should wake up child 2 and so on. This should be repeated ‘n’ times

Code

```c
#include "stdio.h"
#include "stdlib.h"
#include "unistd.h"
#include "wait.h"
#define pr printf
#define sc scanf

int a=30,ch=0,n=3;

pthread_mutex_t mutex;
pid_t A;

int main(){
  A=fork();
  if(A==0){
    for(int i=0;i<n;++i){
      if(fork()==0){
        sleep(2);
        a+=1;
        pr("%d\n",a);
        pr("Child 1 %d\n",getpid());
        sleep(2);
        break;
      }
      else{
        if(fork()==0){
          sleep(2);
          a+=2;
          pr("%d\n",a);
          pr("Child 2 %d\n",getpid());
          sleep(2);
          break;
        }
        else{
          if(fork()==0){
            sleep(3);
            a+=3;
            pr("%d\n",a);
            pr("Child 3 %d\n",getpid());
            sleep(3);
            break;
          }
        }
      }for(int j=0;j<n;++j) wait(0);
      }
  }
  else{
    wait(0);
    pr("Parent\n");
  }
}
```

## Output



```
coder@coder:~/Documents$ gcc fat.c -o out
coder@coder:~/Documents$ ./out
31
Child 1 8398
32
Child 2 8399
33
Child 3 8400
31
Child 1 8401
32
Child 2 8402
33
Child 3 8403
31
Child 1 8406
32
Child 2 8407
33
Child 3 8408
Parent
coder@coder:~/Documents$ 
```