

18BCB0142
David B.A.De Vieira Velho
OS Lab DA3

Q1. Bankers algorithm

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

```
int main()  
{
```

```
int alloc[6][3] = { { 0, 1, 0 }, // P0 // Allocation Matrix  
                   { 2, 0, 0 }, // P1  
                   { 3, 0, 2 }, // P2  
                   { 2, 1, 1 }, // P3  
                   { 0, 0, 2 },  
                   { 0, 1, 3 } }; // P4
```

```
int max[6][3] = { { 7, 5, 3 }, // P0 // MAX Matrix  
                 { 3, 2, 2 }, // P1  
                 { 9, 0, 2 }, // P2  
                 { 2, 2, 2 }, // P3  
                 { 4, 3, 3 },  
                 { 2, 3, 4 } }; // P4
```

```
int available[3] = { 3, 3, 2 }; // Available Resources  
int n, m, i, j, k;  
n = 6; // Number of processes  
m = 3; // Number of 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] > available[j]){  
                    flag = 1;  
                    break;  
                }  
            }  
        }
```

```

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

```

```

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

```

```

return (0);

```

```

}

```

Output:

The screenshot shows a C program in a code editor with a terminal window below it. The code defines 6 processes (P0-P5) and 3 resources (A, B, C). It calculates the maximum resource needs (max) and the available resources. The terminal output shows the required sequence of processes: P1 -> P3 -> P4 -> P5 -> P0 -> P2.

```

main.c
1 #include <stdio.h>
2
3 int main()
4 {
5
6 int alloc[6][3] = { { 0, 1, 0 }, // P0 // A
7                    { 2, 0, 0 }, // P1
8                    { 3, 0, 2 }, // P2
9                    { 2, 1, 1 }, // P3
10                   { 0, 0, 2 },
11                   { 0, 1, 3 } }; // P4
12
13 int max[6][3] = { { 7, 5, 3 }, // P0 // MAX
14                  { 3, 2, 2 }, // P1
15                  { 9, 0, 2 }, // P2
16                  { 2, 2, 2 }, // P3
17                  { 4, 3, 3 },
18                  { 2, 3, 4 } }; // P4
19
20 int available[3] = { 3, 3, 2 }; // Availabl
21
22 int n, m, i, j, k;
23 n = 6; // Number of processes
24 m = 3; // Number of resources
25
26 int f[n], ans[n], ind = 0;
27 for (k = 0; k < n; k++) {
28     f[k] = 0;
29 }
30
31 printf("Following is the required Sequence\n");
32 for (i = 0; i < n - 1; i++)
33     printf(" P%d ->", ans[i]);
34 printf(" P%d\n", ans[n - 1]);
35
36 return (0);
37 }

```

Following is the required Sequence
P1 -> P3 -> P4 -> P5 -> P0 -> P2

...Program finished with exit code 0
Press ENTER to exit console.

Q2. Multiple logins by users

```
#include <stdio.h>
#include <stdlib.h>
int main(){

    system("last | sort | cut -d ' ' -f 1 | uniq -c | sort -n");

    return 0;
}
```

Output :

```
1
1 wtmp
6 david
6 reboot
```

(did not have a linux computer now to record output)

Q3. Bash Program for average grade

```
#!/bin/bash
echo -n "Enter the number of marks: "
read n
i=0

sum=0

for(( i; i < $n; i++ ));
do
    echo -n "Marks: "
    read m
    while [ $m -lt 0 ] || [ $m -gt 100 ]
    do
        echo -n "Enter a positive number: "
        read m
    done

    sum=$((sum+$m))

done

sum=$((sum/$n));
if [ $sum -ge 90 ]
```

```

then
    echo S grade.
    echo $\n'
elif [ $sum -ge 80 ] && [ $sum -lt 90 ]
then
    echo A grade.
    echo $\n'
elif [ $sum -ge 70 ] && [ $sum -lt 80 ]
then
    echo B grade.
    echo $\n'
elif [ $sum -ge 60 ] && [ $sum -lt 70 ]
then
    echo C grade.
    echo $\n'
elif [ $sum -ge 55 ] && [ $sum -lt 60 ]
then
    echo D grade.
    echo $\n'
elif [ $sum -ge 50 ] && [ $sum -lt 55 ]
then
    echo E grade.
    echo $\n'
else
    echo F grade.
    echo $\n'
fi

```

```

49     echo F grade.
50     echo $\n'
51     fi

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```

david@DLinux:~/OSLab/DA3$ bash q3.sh
Enter the number of marks: 4
Marks: 90
Marks: 80
Marks: 70
Marks: 60
B grade.

david@DLinux:~/OSLab/DA3$

```

Q4. Bash script to display files in users home directory

```

#!/bin/bash
DIR=/home/david

```

```

for list in `ls -p /home/david`;
do
    if echo -n $list | grep -v /
    then
        echo `ls -l $DIR/$list | cut -d ' ' -f 1`
        chmod 444 $DIR/$list
        echo 'permission changed'
        echo $"\\n"
    else
        echo $list
    fi
done;
echo $"\\n"
echo "New permissions"
for list in `ls -p /home/david`;
do
    if echo -n $list | grep -v /
    then
        echo `ls -l $DIR/$list | cut -d ' ' -f 1`
        echo $"\\n"
    else
        echo $list
    fi
done;

```

Q5) Pattern

```

n=5

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

    for(( j=i-n; j<n; j++ ))

        echo -ne " ";

    done

    for(( k=1; k<=i; k++ ))
    do
        echo -ne "$k"
    done

    for(( l=i-1; l>=1; l-- ))
    do
        echo -ne "$l"
    done

    echo;

```

done

Output:

```
1 n=5
2
3 for (( i=1; i<=n; i++))
4 do
5
6     for(( j=i-n; j<n; j++))
7     do
8
9         echo -ne " ";
10
11     done
12
13     for(( k=1; k<=i; k++))
14     do
15         echo -ne "$k"
16     done
17
18     for(( l=i-1; l>=1; l--))
19     do
20         echo -ne "$l"
21     done
22
23     echo;
24
25 done
26
```

```
$bash -f main.sh
1
121
12321
1234321
123454321
```

Q6)

```
#!/bin/bash
echo "Select an option:"
echo "a. Print first n triangular numbers"
echo "b. Check if a number is Automorphic Number"
echo "c. Check if a number is Abundant Number"
echo "d. Exit"
while :
do
echo -n 'Select an option: '
read CHAR
case $CHAR in
a)
echo -n "a) Input a number: "
read n
l=0
j=0
for(( i=1; i <= n; i++));
do
l=$((j + $i))
j=l
echo -n ' $l'
done;
echo $'\n'
;;
b)
echo -n 'b) Input a number: '
read n
m=$((n*$n))
```

```

while (( $n != 0 ));
do
check=0
a=$(( $n % 10 ))
b=$(( $m % 10 ))
if [ $a -eq $b ];
then
n=$(( $n / 10 ))
m=$(( $m / 10 ))
check=1
else
check=0
break
fi
done
if [ $check -eq 1 ];
then
echo "Automorphic"
elif [ $check -eq 0 ];
then
echo "Not Automorphic"
fi
;;
c)
sum=0
echo -n "c) Input a number: "
read n
for(( i=1; i < n; i++ ));
do
m=$(( $n % $i ))
echo $m
if [ $m -eq 0 ];
then
sum=$(( $sum + $i ))
fi
done
echo $'\n'
echo $sum
if [ $sum -gt $n ];
then
echo 'Abundant number'
else
echo 'Not an Abundant number'
fi
;;
*)
echo "Enter a valid option"
break
;;
esac
done;

```

Output: (no ubuntu laptop at the time to record output)

Select an option:

- a. Print first n triangular numbers
- b. Check if a number is Automorphic Number
- c. Check if a number is Abundant Number
- d. Exit

Select an option: a

a) Input a number: 8

1 3 6 10 15 21 28 36

Select an option: b

b) Input a number: 76

Automorphic

Select an option: c

c) Input a number: 12

Abundant number