Name: Karan Salunkhe

SE-A, Roll No-23173

EXPERIMENT NO. 4

#include <iostream>

using namespace std;

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;

}

}

}

}

cout << "Program By KARAN SALUNKHE, SE-A, Roll No-23173" << endl;

cout << "Following is the SAFE Sequence" << endl;

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

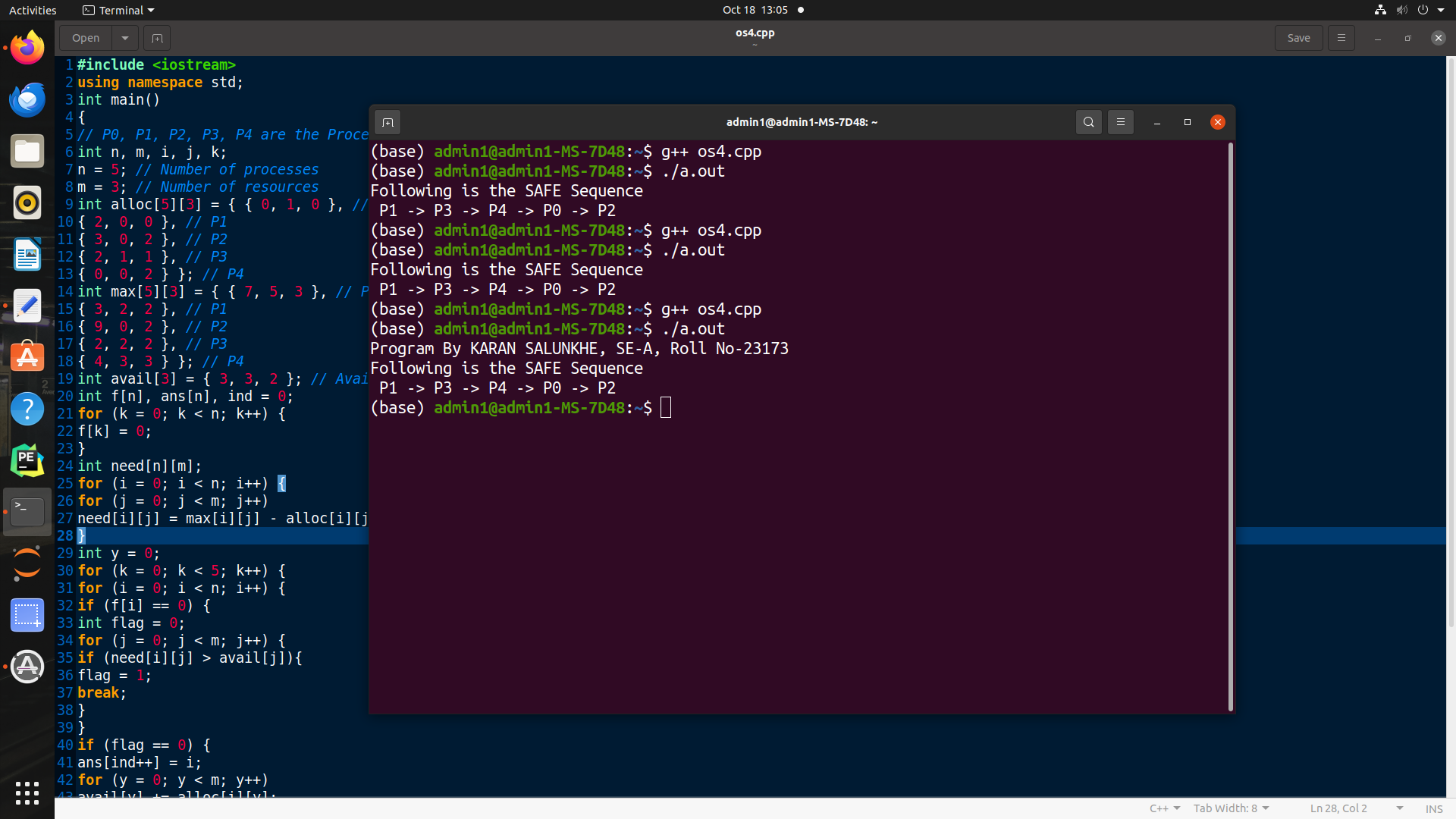
cout << " P" << ans[i] << " ->";

cout << " P" << ans[n - 1] <<endl;

return (0);

}

OUTPUT



Name: Karan Salunkhe

SE-A, Roll No-23173

EXPERIMENT NO. 5

#include<stdio.h>

int main() {

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

printf("Program By - KARAN SALUNKHE, SE-A, Roll No-23173\n");

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]);

}

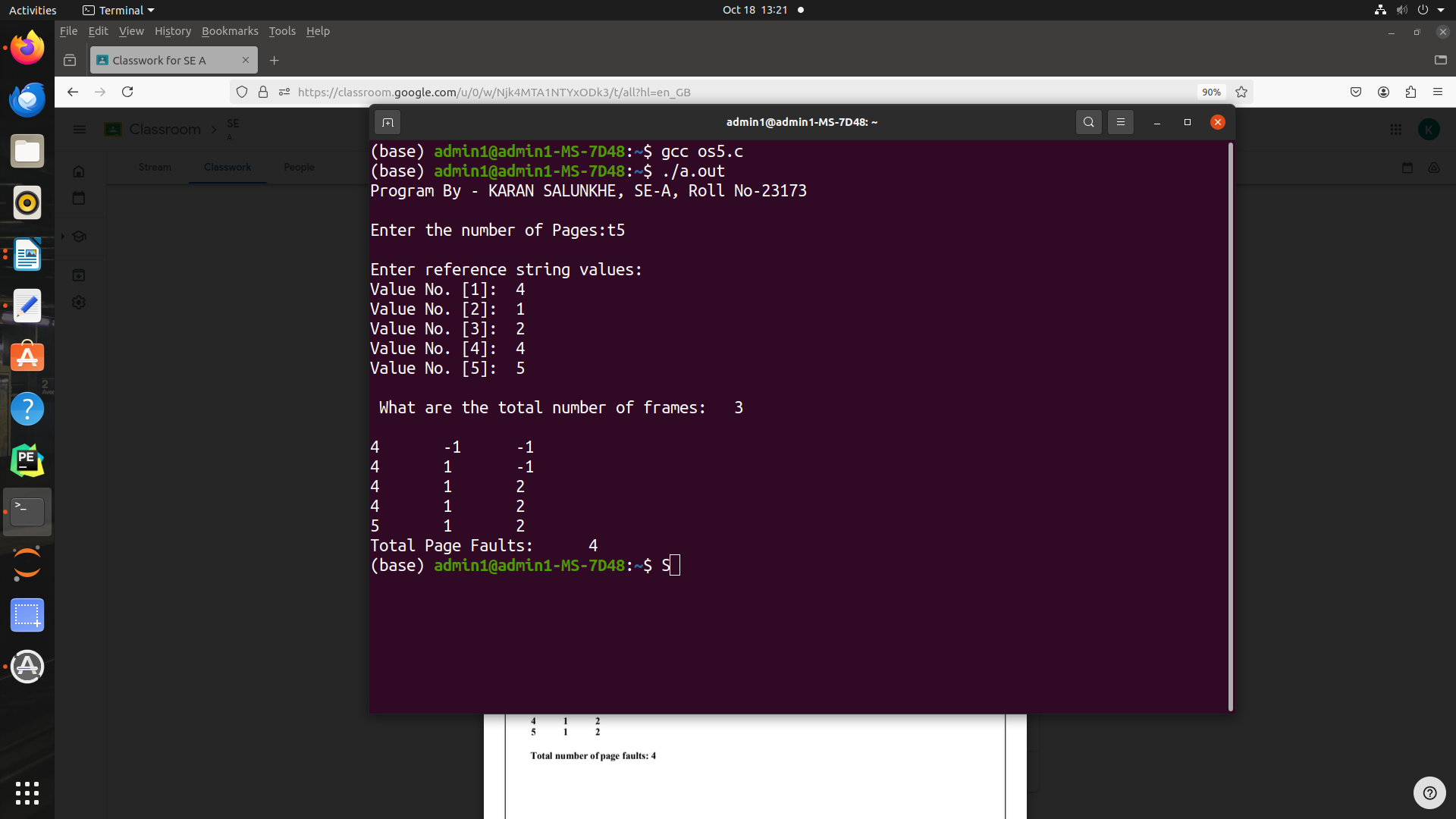
}

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

return 0;

}

OUTPUT



Name: Karan Salunkhe

SE-A, Roll No-23173

EXPERIMENT NO. 6

#include<iostream>

#include<bits/stdc++.h>

using namespace std;

// Function to find page faults using indexes

int pageFaults(int pages[], int n, int capacity)

{

// To represent set of current pages. We use

// an unordered\_set so that we quickly check

// if a page is present in set or not

unordered\_set<int> s;

// To store least recently used indexes

// of pages.

unordered\_map<int, int> indexes;

// Start from initial page

int page\_faults = 0;

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

{

// Check if the set can hold more pages

if (s.size() < capacity)

{

// Insert it into set if not present

// already which represents page fault

if (s.find(pages[i])==s.end())

{

s.insert(pages[i]);

// increment page fault

page\_faults++;

}

// Store the recently used index of

// each page

indexes[pages[i]] = i;

}

// If the set is full then need to perform lru

// i.e. remove the least recently used page

// and insert the current page

else

{

// Check if current page is not already

// present in the set

if (s.find(pages[i]) == s.end())

{

// Find the least recently used pages

// that is present in the set

int lru = INT\_MAX, val;

for (auto it=s.begin(); it!=s.end(); it++)

{

if (indexes[\*it] < lru)

{

lru = indexes[\*it];

val = \*it;

}

}

// Remove the indexes page

s.erase(val);

// insert the current page

s.insert(pages[i]);

// Increment page faults

page\_faults++;

}

// Update the current page index

indexes[pages[i]] = i;

}

}

return page\_faults;

}

// Driver code

int main()

{

cout<<"Program By - KARAN SALUNKHE, SE-A, Roll No - 23173\n";

int pages[] = {7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2};

int n = sizeof(pages)/sizeof(pages[0]);

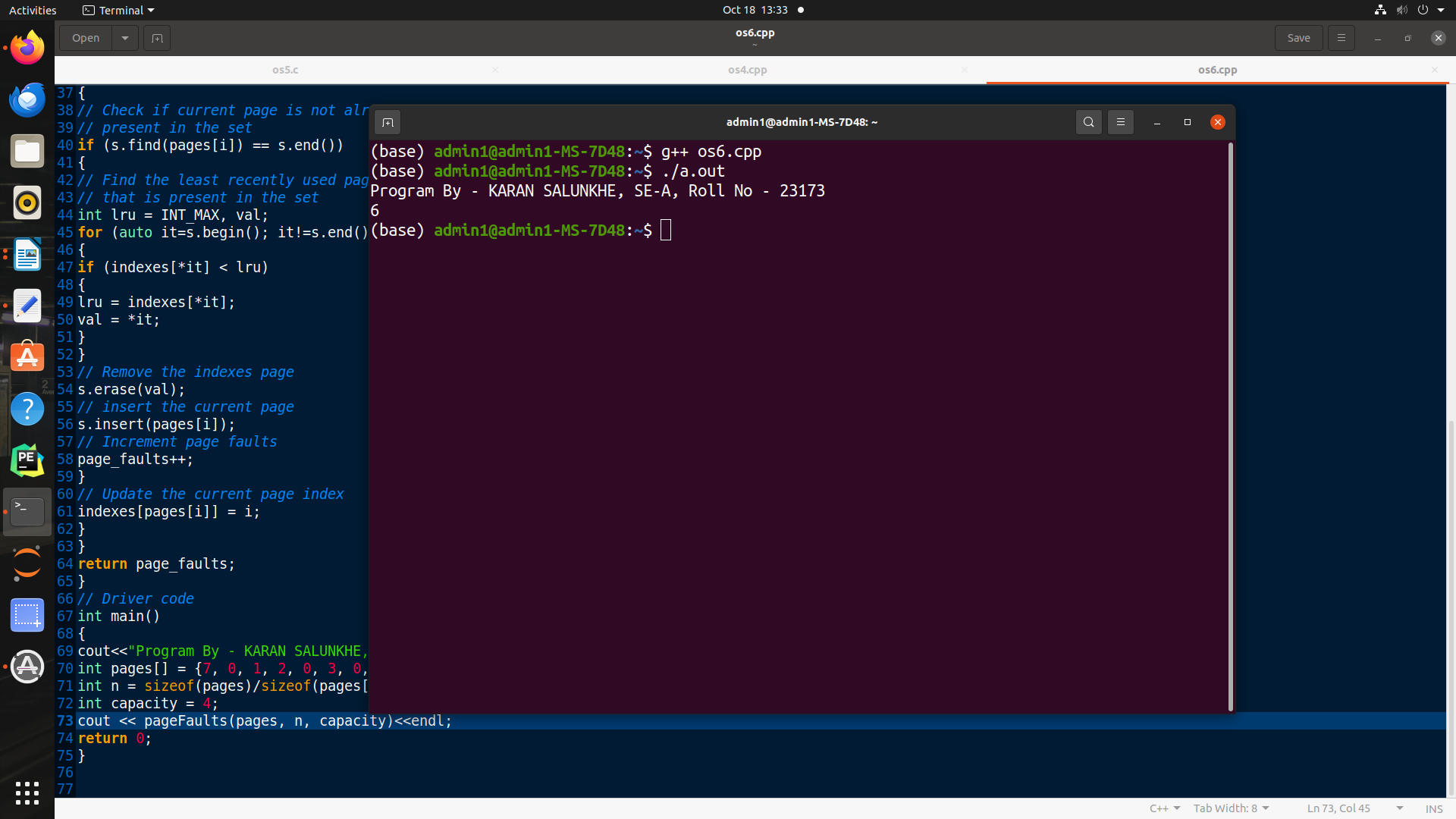
int capacity = 4;

cout << pageFaults(pages, n, capacity)<<endl;

return 0;

}

OUTPUT



Name: Karan Salunkhe

SE-A, Roll No-23173

EXPERIMENT NO. 7

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

int main()

{

//A null terminated array of character

//pointers

printf("Program By - KARAN SALUNKHE, SE-A, Roll No - 23173\n");

char \*args[]={"./EXEC",NULL};

execvp(args[0],args);

/\*All statements are ignored after execvp() call as this whole

process(execDemo.c) is replaced by another process (EXEC.c)

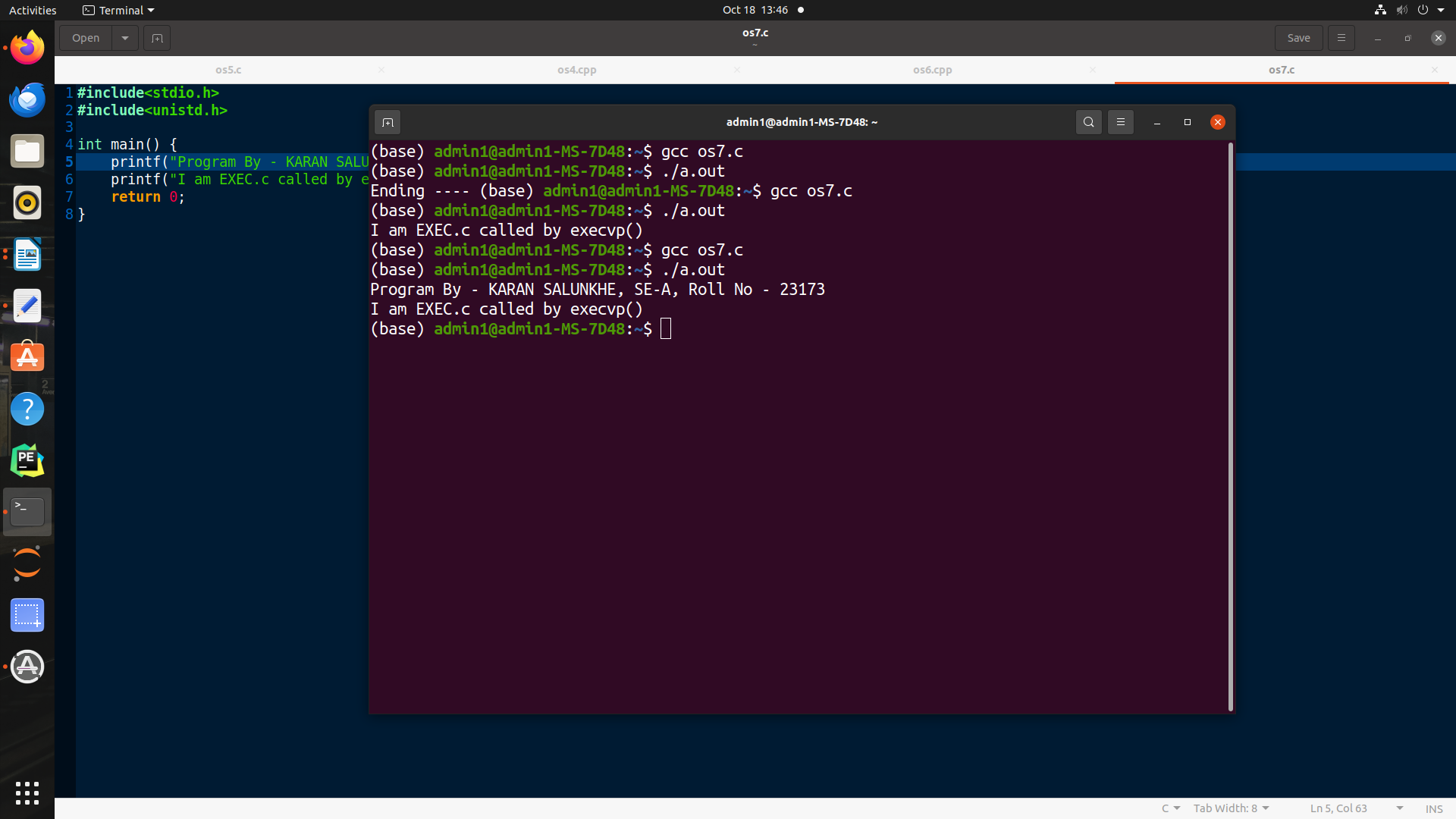
\*/

printf("Ending ---- ");

return 0;

}

OUTPUT



Name: Karan Salunkhe

SE-A, Roll No-23173

EXPERIMENT NO. 8

Opt=1

while [ “$opt” -lt 6 ]

do

echo -e “Choose one of the Following\n1. Create a New Address Book\n2. View

records\n3. Insert new Record\n4. Delete a Record\n5. Modify a Record\n6. Exit”

# echo -e, enables special features of echo to use \n \t \b etc.

read opt

case $opt in

1)

echo “Enter filename”

read fileName

if [ -e $fileName ] ; then # -e to check if file exists, if exits remove the file

rm $fileName

fi

Cont=1

echo

“NAME\tNUMBER\t\tADDRESS\n===============================\n” | cat >>$fileName

while [ “$cont” -gt 0 ]

do

echo “\nEnter Name”

read name

echo “Enter Phone Number of $name”

read number

echo “Enter Address of $name”

read address

echo “$name\t$number\t\t$address” | cat >> $fileName

echo “Enter 0 to Stop, 1 to Enter next”

read cont

done

;;

2)

cat $fileName

;;

3)

echo “\nEnter Name”

read name

echo “Enter Phone Number of $name”

read number

echo “Enter Address of $name”

read address

echo “$name\t$number\t\t$address” | cat >> $fileName

;;

4)

echo “Delete record\nEnter Name/Phone Number”

read pattern

Temp=”temp”

grep -v $pattern $fileName | cat >> $temp

rm $fileName

cat $temp | cat >> $fileName

rm $temp;;

5)

echo “Modify record\nEnter Name/Phone Number”

read pattern

Temp=”temp”

grep -v $pattern $fileName | cat >> $temp

rm $fileName

cat $temp | cat >> $fileName

rm $temp

echo “Enter Name”

read name

echo “Enter Phone Number of $name”

read number

echo “Enter Address of $name”

read address

echo -e “$name\t$number\t$address” | cat >> $fileName;;

esac

done

Name: Karan Salunkhe

SE-A, Roll No-23173

EXPERIMENT NO. 9

echo "Enter Two Numbers"

read a b

echo "What do you want to do? (1 to 5)"

echo "1) Sum"

echo "2) Difference"

echo "3) Product"

echo "4) Quotient"

echo "5) Remainder"

echo "Enter your Choice"

read n

case "$n" in

1) echo "The Sum of $a and $b is `expr $a + $b`";;

2) echo “The Difference between $a and $b is `expr $a - $b`”;;

3) echo “The Product of the $a and $b is `expr $a \\* $b`”;;

4) echo “The Quotient of $a by $b is `expr $a / $b`”;;

5) echo “The Remainder of $a by $b is `expr $a % $b`”;;

esac

Name: Karan Salunkhe

SE-A, Roll No-23173

EXPERIMENT NO. 10

#include<stdio.h>

#include<unistd.h>

int main() {

printf("Program By - KARANS SALUNKHE, SE-A, ROll No - 23173");

int pipefds1[2], pipefds2[2];

int returnstatus1, returnstatus2;

int pid;

char pipe1writemessage[20] = "Hi";

char pipe2writemessage[20] = "Hello";

char readmessage[20];

returnstatus1 = pipe(pipefds1);

if (returnstatus1 == -1) {

printf("Unable to create pipe 1 \n");

return 1;

}

returnstatus2 = pipe(pipefds2);

if (returnstatus2 == -1) {

printf("Unable to create pipe 2 \n");

return 1;

}

pid = fork();

if (pid != 0) // Parent process

{

close(pipefds1[0]); // Close the unwanted pipe1 read side

close(pipefds2[1]); // Close the unwanted pipe2 write side

printf("In Parent: Writing to pipe 1 – Message is %s\n", pipe1writemessage);

write(pipefds1[1], pipe1writemessage, sizeof(pipe1writemessage));

read(pipefds2[0], readmessage, sizeof(readmessage));

printf("In Parent: Reading from pipe 2 – Message is %s\n", readmessage);

}

else

{

//child process

close(pipefds1[1]); // Close the unwanted pipe1 write side

close(pipefds2[0]); // Close the unwanted pipe2 read side

read(pipefds1[0], readmessage, sizeof(readmessage));

printf("In Child: Reading from pipe 1 – Message is %s\n", readmessage);

printf("In Child: Writing to pipe 2 – Message is %s\n", pipe2writemessage);

write(pipefds2[1], pipe2writemessage, sizeof(pipe2writemessage));

}

return 0;

}

OUTPUT

