Experiment 4

Aim: Write a program to implement the Bankers Algorithm

Roll No: SEAD23155

Class: SE-A

Batch: A4

```
#include <iostream>
using namespace std;

int main() {
    int n, m, i, j, k;
    n = 5; // Number of processes
    m = 3; // Number of resources

    int alloc[5][3] = {
        { 0, 1, 0 },
        { 2, 0, 0 },
        { 3, 0, 2 },
        { 2, 1, 1 },
        { 0, 0, 2 }
    };

    int max[5][3] = {
        { 7, 5, 3 },
        { 3, 2, 2 },
        { 9, 0, 2 },
        { 2, 2, 2 },
        { 4, 3, 3 }
    };
```

```
int avail[3] = { 3, 3, 2 };
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 << "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:

```
saishhh@Ubuntu-Dabba:~$ g++ exp5.cpp
saishhh@Ubuntu-Dabba:~$ ./a.out
Following is the SAFE Sequence
 P1 -> P3 -> P4 -> P0 -> P2
saishhh@Ubuntu-Dabba:~$
```

Experiment 5

Aim: Write a program to implement page Replacement strategies (FIFO, LRU,Optimal)

Roll No: SEAD23155

Class: SE-A

Batch: A4

FIFO Page Replacement Algorithm

```
#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("\nWhat 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:

```
saishhh@Ubuntu-Dabba:~$ g++ exp5.cpp
saishhh@Ubuntu-Dabba:~$ ./a.out

Enter the number of Pages:      6

Enter reference string values:
Value No. [1]:  1
Value No. [2]:  3
Value No. [3]:  0
Value No. [4]:  3
Value No. [5]:  5
Value No. [6]:  6

What are the total number of frames:    3

1       -1      -1
1       3       -1
1       3       0
1       3       0
5       3       0
5       6       0
Total Page Faults:      5
saishhh@Ubuntu-Dabba:~$ Name: Saish Baviskar, Roll No: 23155, Batch: A4
```

LRU (Least Recently Used) Page Replacement Algorithm

```
#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;
    int 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:

```
saishhh@Ubuntu-Dabba:~$ g++ exp5.cpp
saishhh@Ubuntu-Dabba:~$ ./a.out
Enter number of frames: 3
Enter number of pages: 7
Enter reference string: 2 3 2 1 5 2 4

2       -1      -1
2       3       -1
2       3       -1
2       3       1
2       5       1
2       5       1
2       5       4

Total Page Faults = 5saishhh@Ubuntu-Dabba:~$ Name: Saish Baviskar, Roll No: 23155,  Batch: A4
```

Optimal Page Replacement Algorithm

```
#include <stdio.h>

int main() {
    int frames_number, pages_number, frames[10], pages[30], temp[10];
    int flag1, flag2, flag3, i, j, k, pos, max, miss = 0;
```

```
printf("Enter number of frames: ");
scanf("%d", &frames_number);

printf("Enter number of pages: ");
scanf("%d", &pages_number);

printf("Enter page reference string: ");
for (i = 0; i < pages_number; ++i) {
    scanf("%d", &pages[i]);
}

for (i = 0; i < frames_number; ++i) {
    frames[i] = -1;
}

for (i = 0; i < pages_number; ++i) {
    flag1 = flag2 = 0;

    for (j = 0; j < frames_number; ++j) {
        if (frames[j] == pages[i]) {
            flag1 = flag2 = 1;
            break;
        }
    }

    if (flag1 == 0) {
        for (j = 0; j < frames_number; ++j) {
            if (frames[j] == -1) {
                frames[j] = pages[i];
                flag2 = 1;
```

```
            miss++;
            break;
        }
    }
}

if (flag2 == 0) {
    flag3 = 0;
    for (j = 0; j < frames_number; ++j) {
        temp[j] = -1;
        for (k = i + 1; k < pages_number; ++k) {
            if (frames[j] == pages[k]) {
                temp[j] = k;
                break;
            }
        }
    }

    for (j = 0; j < frames_number; ++j) {
        if (temp[j] == -1) {
            pos = j;
            flag3 = 1;
            break;
        }
    }

    if (flag3 == 0) {
        max = temp[0];
        pos = 0;
        for (j = 1; j < frames_number; ++j) {
```

```
            if (temp[j] > max) {
                max = temp[j];
                pos = j;
            }
        }
    }

    frames[pos] = pages[i];
    miss++;
  }

  printf("\n");
  for (j = 0; j < frames_number; ++j) {
    printf("%d\t", frames[j]);
  }
 }

 printf("\n\nTotal Page miss = %d", miss);
 return 0;
}
```

OUTPUT:

```
saishhh@Ubuntu-Dabba:~$ g++ exp5.cpp
saishhh@Ubuntu-Dabba:~$ ./a.out
Enter number of frames: 3
Enter number of pages: 7
Enter page reference string: 4 7 6 1 7 6 1

4       -1      -1
4       7       -1
4       7       6
1       7       6
1       7       6
1       7       6
1       7       6

Total Page miss = 4saishhh@Ubuntu-Dabba:~$ Name: Saish Baviskar, Roll No: 23155, Batch: A4
```

Experiment 6

Aim: Write a Program to implement paging simulation using Least Recently Used (LRU) and Optimal algorithm

Roll No: SEAD23155

Class: SE-A

Batch: A4

LRU Page Replacement Algorithm

```
#include <bits/stdc++.h>
using namespace std;

int pageFaults(int pages[], int n, int capacity) {
    unordered_set<int> s;
```

```
    unordered_map<int, int> indexes;
    int page_faults = 0;

    for (int i = 0; i < n; i++) {
        if (s.size() < capacity) {
            if (s.find(pages[i]) == s.end()) {
                s.insert(pages[i]);
                page_faults++;
            }
            indexes[pages[i]] = i;
        } else {
            if (s.find(pages[i]) == s.end()) {
                int lru = INT_MAX, val;
                for (auto it = s.begin(); it != s.end(); it++) {
                    if (indexes[*it] < lru) {
                        lru = indexes[*it];
                        val = *it;
                    }
                }
                s.erase(val);
                s.insert(pages[i]);
                page_faults++;
            }
            indexes[pages[i]] = i;
        }
    }
    return page_faults;
}

int main() {
```

```
    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);
    return 0;
}
```

OUTPUT:

```
saishhh@Ubuntu-Dabba:~$ g++ exp5.cpp
^[[Asaishhh@Ubuntu-Dabba:~$ ./a.out
6
saishhh@Ubuntu-Dabba:~$ Name: Saish Baviskar, Roll No: 23155, Batch: A4
```

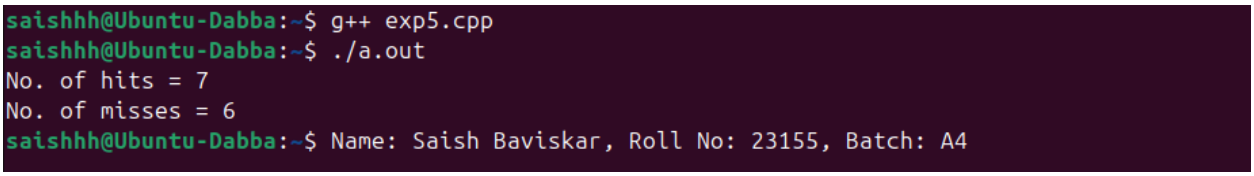
Optimal Page Replacement Algorithm

```
#include <bits/stdc++.h>
using namespace std;

bool search(int key, vector<int>& fr) {
    for (int i = 0; i < fr.size(); i++)
        if (fr[i] == key)
            return true;
    return false;
}

int predict(int pg[], vector<int>& fr, int pn, int index) {
    int res = -1, farthest = index;
    for (int i = 0; i < fr.size(); i++) {
        int j;
        for (j = index; j < pn; j++) {
```

```
            if (fr[i] == pg[j]) {
                if (j > farthest) {
                    farthest = j;
                    res = i;
                }
                break;
            }
        }
        if (j == pn)
            return i;
    }
    return (res == -1) ? 0 : res;
}

void optimalPage(int pg[], int pn, int fn) {
    vector<int> fr;
    int hit = 0;
    for (int i = 0; i < pn; i++) {
        if (search(pg[i], fr)) {
            hit++;
            continue;
        }
        if (fr.size() < fn)
            fr.push_back(pg[i]);
        else {
            int j = predict(pg, fr, pn, i + 1);
            fr[j] = pg[i];
        }
    }
    cout << "No. of hits = " << hit << endl;
```

```
    cout << "No. of misses = " << pn - hit << endl;
}

int main() {
    int pg[] = {7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2};
    int pn = sizeof(pg) / sizeof(pg[0]);
    int fn = 4;
    optimalPage(pg, pn, fn);
    return 0;
}
```

OUTPUT:



Experiment 7

Aim: Implement UNIX system calls like ps, fork, join, exec family, and wait for process management (use shell script/ Java/ C programming)Shell programming

Roll No: SEAD23155

Class: SE-A

Batch: A4

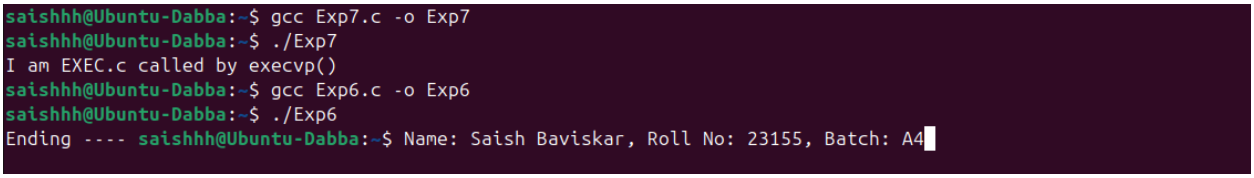
//EXEC.c

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

int main() {
    printf("I am EXEC.c called by execvp()\n");
    return 0;
}
```

//execDemo.c

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

int main() {
    // A null terminated array of character pointers
    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:



Experiment 8

Aim: Write a program to implement an address book with options given below: a) Create address book. b) View address book. c) Insert a record. d) Delete a record. e) Modify a record. f) Exit

Roll No: SEAD23155

Class: SE-A

Batch: A4

```
#!/bin/bash

it=0  # Initialize iterator variable
op=0  # Initialize 'op' variable to avoid comparison errors

while [ $op -lt 7 ]
do
  echo "Enter the option"
  echo "1 for create"
  echo "2 for add"
  echo "3 for display"
  echo "4 for search"
  echo "5 for delete"
  echo "6 for modify"
  echo "7 for exit"

  echo "Enter your choice:"
```

```
read op

case "$op" in
  1)
    echo "Enter the name for the database:"
    read db
    touch "$db"
    ;;

  2)
    echo "In which database do you want to add records?"
    read db
    echo "Enter the number of records:"
    read n

    while [ $it -lt $n ]
    do
      echo "Enter ID:"
      read id1

      echo "Enter name:"
      read nm

      echo "Enter address:"
      read add
      pa="^[A-Za-z0-9 ]+$"
      while [[ ! $add =~ $pa ]]
      do
        echo "Enter a valid address:"
        read add
```

```
    done

    echo "Enter phone number (10 digits):"
    read ph
    pat="^[0-9]{10}$"
    while [[ ! $ph =~ $pat ]]
    do
      echo "Please enter a valid phone number (10 digits):"
      read ph
    done

    echo "Enter email:"
    read em
    patem="^[a-z0-9._%-]+@[a-z]+\.[a-z]{2,4}$"
    while [[ ! $em =~ $patem ]]
    do
      echo "Please enter a valid email address:"
      read em
    done

    echo "$id1,$nm,$add,$ph,$em" >> "$db"
    it=$((it + 1))
    echo "$it record(s) entered"
  done
  ;;

3)
  echo "Enter the name of the database to display:"
  read db
  cat "$db"
```

```
  ;;

4)
  echo "Enter the name of the database to search:"
  read db
  echo "Enter email to search:"
  read em1
  if grep -q "$em1" "$db"; then
    echo "Record found"
  else
    echo "Not found"
  fi
  ;;

5)
  echo "Enter the name of the database:"
  read db
  echo "Enter ID to delete:"
  read id1
  echo "Enter line number to delete:"
  read linenumber

  for line in $(grep -n "$id1" "$db")
  do
    number=$(echo "$line" | cut -d: -f1)
    if [ "$number" == "$linenumber" ]; then
      sed -i "${linenumber}d" "$db"
      echo "Record removed"
    fi
  done
```

```
    ;;

  6)
    echo "Enter the name of the database:"
    read db
    echo "Enter ID to modify:"
    read id1
    echo "Enter line number to modify:"
    read linenumber

    for line in $(grep -n "$id1" "$db")
    do
      number=$(echo "$line" | cut -d: -f1)
      if [ "$number" == "$linenumber" ]; then
        echo "What would you like to change (id,name,address,mobile,email)?"
        read edit
        sed -i "${linenumber}s/.*/$edit/" "$db"
        echo "Record edited"
      fi
    done
fi
“7”)
Echo “bye”
;;
*)echo invalid input
Esac
Done
```

OUTPUT:

```
saishhh@Ubuntu-Dabba:~$ ./database.sh
Enter the option
1 for create
2 for add
3 for display
4 for search
5 for delete
6 for modify
7 for exit
Enter your choice:
1
Enter the name for the database:
students.db
Enter the option
1 for create
2 for add
3 for display
4 for search
5 for delete
6 for modify
7 for exit
Enter your choice:
2
In which database do you want to add records?
students.db
Enter the number of records:
2
Enter ID:
101
Enter name:
Alice
Enter address:
123 Main St
Enter phone number (10 digits):
9876543210
Enter email:
alice@example.com
1 record(s) entered
```

```
1 record(s) entered
Enter ID:
102
Enter name:
Bob
Enter address:
456 Elm St
Enter phone number (10 digits):
1234567890
Enter email:
bob@example.com
2 record(s) entered
Enter the option
1 for create
2 for add
3 for display
4 for search
5 for delete
6 for modify
7 for exit
Enter your choice:
3
Enter the name of the database to display:
students.db
101,Alice,123 Main St,7387328425,alice@example.com
101,Alice,123 Main St,9876543210,alice@example.com
102,Bob,456 Elm St,1234567890,bob@example.com
Enter the option
1 for create
2 for add
3 for display
4 for search
5 for delete
6 for modify
7 for exit
Enter your choice:
4
```

```
Enter your choice:
4
Enter the name of the database to search:
students.db
Enter email to search:
alice@example.com
Record found
Enter the option
1 for create
2 for add
3 for display
4 for search
5 for delete
6 for modify
7 for exit
Enter your choice:
5
Enter the name of the database:
students.db
Enter ID to delete:
102
Enter line number to delete:
2
Enter the option
1 for create
2 for add
3 for display
4 for search
5 for delete
6 for modify
7 for exit
Enter your choice:
6
Enter the name of the database:
students.db
Enter ID to modify:
101
```

```
Enter the option
1 for create
2 for add
3 for display
4 for search
5 for delete
6 for modify
7 for exit
Enter your choice:
7
Bye!
saishhh@Ubuntu-Dabba:~$ Name: Saish Baviskar, Roll No: 23155, Batch: A4
```

Experiment 9

Aim: Create a shell program to do mathematical operations.

Roll No: SEAD23155

Class: SE-A

Batch: A4

```
#!/bin/bash

# Step 2: Read the two numbers
echo "Enter Two Numbers:"
read a b

# Step 3: Get the operation choice from the user
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

# Step 4: Perform the selected operation
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 $a and $b is `expr $a \* $b`";;
  4)
    if [ $b -ne 0 ]; then
```

```
    echo "The Quotient of $a by $b is `expr $a / $b`"
   else
    echo "Error: Division by zero is not allowed"
   fi
   ;;
  5)
   if [ $b -ne 0 ]; then
    echo "The Remainder of $a by $b is `expr $a % $b`"
   else
    echo "Error: Division by zero is not allowed"
   fi
   ;;
  *) echo "Invalid Choice";;
Esac
```

OUTPUT:

```
saishhh@Ubuntu-Dabba:~$ ./calculator.sh
Enter Two Numbers:
10 5
What do you want to do? (1 to 5)
1) Sum
2) Difference
3) Product
4) Quotient
5) Remainder
Enter your Choice:
1
The Sum of 10 and 5 is 15
saishhh@Ubuntu-Dabba:~$ ./calculator.sh
Enter Two Numbers:
10 5
What do you want to do? (1 to 5)
1) Sum
2) Difference
3) Product
4) Quotient
5) Remainder
Enter your Choice:
2
The Difference between 10 and 5 is 5
saishhh@Ubuntu-Dabba:~$ ./calculator.sh
Enter Two Numbers:
10 5
What do you want to do? (1 to 5)
1) Sum
2) Difference
3) Product
4) Quotient
5) Remainder
Enter your Choice:
3
The Product of 10 and 5 is 50
```

```
saishhh@Ubuntu-Dabba:~$ ./calculator.sh
Enter Two Numbers:
10 5
What do you want to do? (1 to 5)
1) Sum
2) Difference
3) Product
4) Quotient
5) Remainder
Enter your Choice:
4
The Quotient of 10 by 5 is 2
saishhh@Ubuntu-Dabba:~$ ./calculator.sh
Enter Two Numbers:
10 5
What do you want to do? (1 to 5)
1) Sum
2) Difference
3) Product
4) Quotient
5) Remainder
Enter your Choice:
5
The Remainder of 10 by 5 is 0
saishhh@Ubuntu-Dabba:~$ Name: Saish Baviskar, Roll No: 23155, Batch: A4
```

Experiment 10

Aim: Create a shell program to do mathematical operations.

Roll No: SEAD23155

Class: SE-A

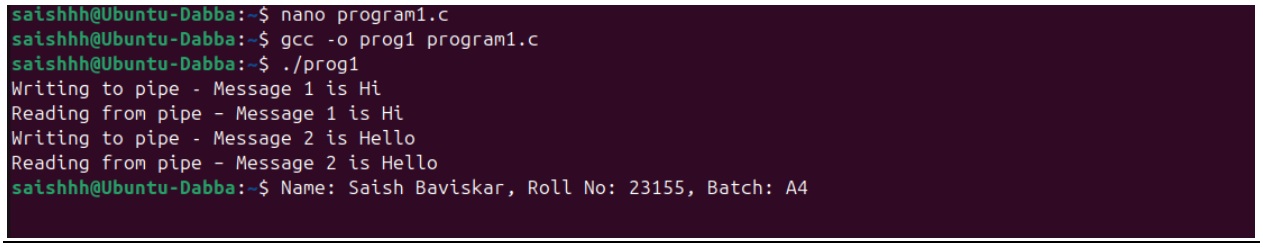
Batch: A4

Program to write and read two messages using pipe.

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

int main() {
    int pipefds[2];
    int returnstatus;
    char writemessages[2][20] = {"Hi", "Hello"};
    char readmessage[20];

    returnstatus = pipe(pipefds);
    if (returnstatus == -1) {
        printf("Unable to create pipe\n");
        return 1;
    }

    printf("Writing to pipe - Message 1 is %s\n", writemessages[0]);
    write(pipefds[1], writemessages[0], sizeof(writemessages[0]));
    read(pipefds[0], readmessage, sizeof(readmessage));
    printf("Reading from pipe – Message 1 is %s\n", readmessage);

    printf("Writing to pipe - Message 2 is %s\n", writemessages[1]);
```

```
    write(pipefds[1], writemessages[1], sizeof(writemessages[1]));
    read(pipefds[0], readmessage, sizeof(readmessage));
    printf("Reading from pipe – Message 2 is %s\n", readmessage);

    return 0;
}
```

OUTPUT:



Program to write and read two messages through the pipe using the parent and the child processes.

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

int main() {
    int pipefds[2];
    int returnstatus;
    int pid;
    char writemessages[2][20] = {"Hi", "Hello"};
    char readmessage[20];

    returnstatus = pipe(pipefds);
    if (returnstatus == -1) {
```

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

    pid = fork();

    if (pid == 0) {
        read(pipefds[0], readmessage, sizeof(readmessage));
        printf("Child Process - Reading from pipe – Message 1 is %s\n", readmessage);
        read(pipefds[0], readmessage, sizeof(readmessage));
        printf("Child Process - Reading from pipe – Message 2 is %s\n", readmessage);
    } else {
        printf("Parent Process - Writing to pipe - Message 1 is %s\n", writemessages[0]);
        write(pipefds[1], writemessages[0], sizeof(writemessages[0]));
        printf("Parent Process - Writing to pipe - Message 2 is %s\n", writemessages[1]);
        write(pipefds[1], writemessages[1], sizeof(writemessages[1]));
    }

    return 0;
}
```

OUTPUT:

```
saishhh@Ubuntu-Dabba:~$ gcc -o prog2 program2.c
saishhh@Ubuntu-Dabba:~$ ./prog2
Parent Process - Writing to pipe - Message 1 is Hi
Parent Process - Writing to pipe - Message 2 is Hello
Child Process - Reading from pipe – Message 1 is Hi
Child Process - Reading from pipe – Message 2 is Hello
saishhh@Ubuntu-Dabba:~$ Name: Saish Baviskar, Roll No: 23155, Batch: A4
```

Two-Way Communication Using Pipes

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

```
#include <unistd.h>

int main() {
    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) {
        close(pipefds1[0]);
        close(pipefds2[1]);
        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 {
        close(pipefds1[1]);
        close(pipefds2[0]);
        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: