Name: Addisu Amare

Reg no:18BML0104

Course name: data structure and algorithm

Course code:CSE2003

Lab slot:L31/32

Assessment:3

1,Create an unordered linked list to enroll the students who wish to participate for a gaming event by taking details like Name, Register No., Age, Phone number. Ensure that no more than five members are there in the list with same age. Perform insertion(), deletion() and display() operations on the Linked List

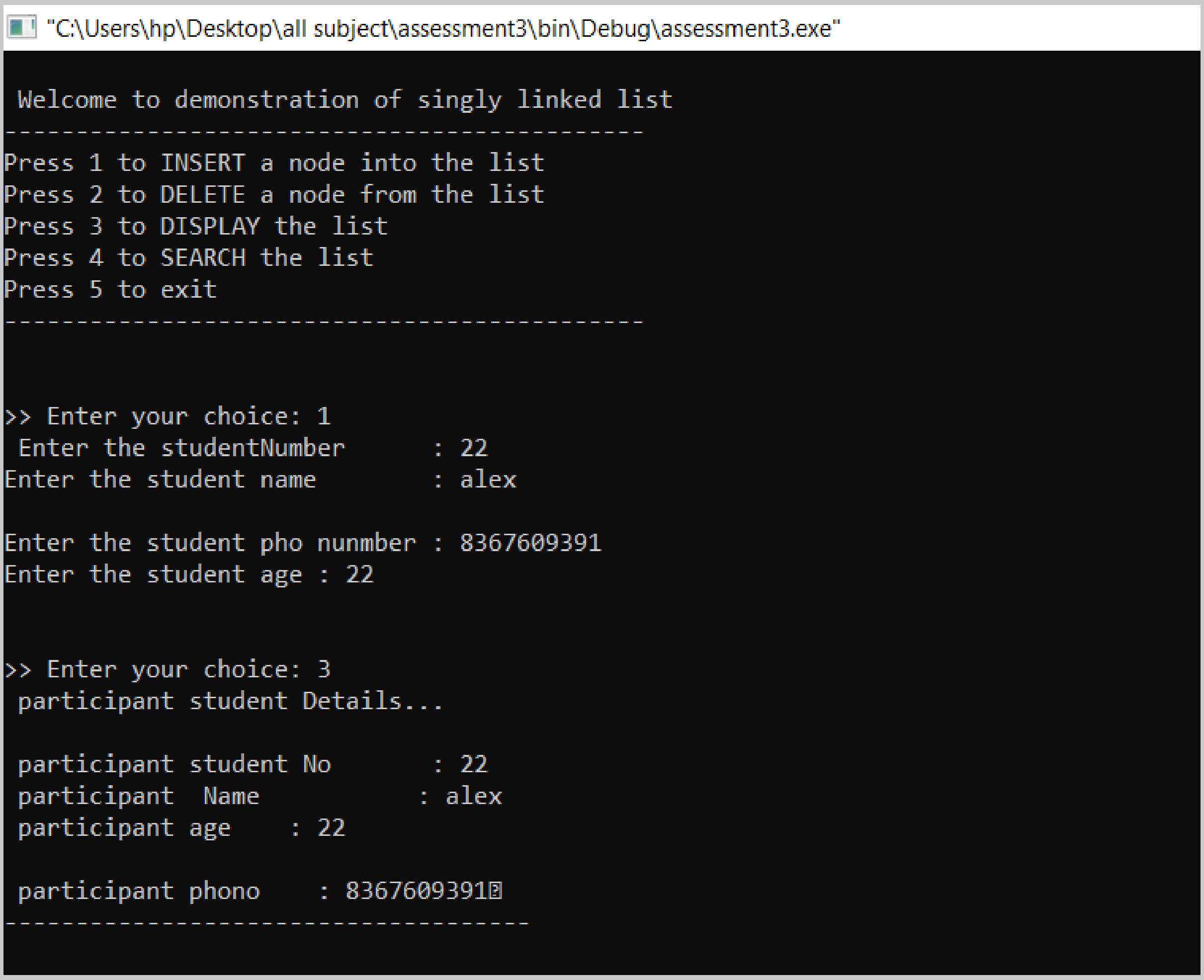
Code :

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX 5
struct stud_data
{
   int  studentno;
   char sName[MAX];
   float age;
   int phono;
   struct stud_data *next;
};
struct stud_data *insert(struct stud_data *front, int id, char name[],
float age,int phono)
{
   struct stud_data *newnode;
   newnode = (struct stud_data*)malloc(sizeof(struct stud_data));
```

```
    if (newnode == NULL)
    {
        printf("\n Allocation failed \n");
        exit(2);
    }
    newnode->studentno = id;
    strcpy(newnode->sName, name);
    newnode->age=age;
    newnode->phono=phono;
    newnode->next = front;
    front = newnode;
    return(front);
}
void printNode(struct stud_data *p)
{
    printf("\n student Details...\n");
    printf("\n student No       : %d", p->studentno);
    printf("\n Name           : %s", p->sName);
    printf("\n age    : %f\n", p->age);
     printf("\n phono    : %d\n", p->phono);
    printf("-------------------------------------\n");
}
struct stud_data* deleteNode(struct stud_data *front, int id)
{
    struct stud_data *ptr;
    struct stud_data *bptr;

    if (front->studentno == id)
    {
        ptr = front;
        printf("\n Node deleted:");
        printNode(front);
        front = front->next;
        free(ptr);
        return(front);
    }
    for (ptr = front->next, bptr = front; ptr != NULL; ptr = ptr->next,
bptr = bptr->next)
    {
        if (ptr->studentno == id)
        {
            printf("\n Node deleted:");
            printNode(ptr);
            bptr->next = ptr->next;
            free(ptr);
            return(front);
        }
    }
    printf("\n student Number %d not found ", id);
    return(front);
}
void search(struct stud_data *front, int key)
{
    struct stud_data *ptr;

    for (ptr = front; ptr != NULL; ptr = ptr -> next)
    {
        if (ptr->studentno == key)
        {
            printf("\n Key found:");
            printNode(ptr);
            return;
        }
    }
    printf("\n registrationNumber %d not found ", key);
}
void display(struct stud_data  *front)
{
    struct stud_data *ptr;

    for (ptr = front; ptr != NULL; ptr = ptr->next)
    {
        printNode(ptr);
    }
}
void menu()
{
    printf("---------------------------------------------\n");
    printf("Press 1 to INSERT a node into the list       \n");
    printf("Press 2 to DELETE a node from the list       \n");
    printf("Press 3 to DISPLAY the list                  \n");
    printf("Press 4 to SEARCH the list                   \n");
    printf("Press 5 to exit                              \n");
    printf("---------------------------------------------\n");
}
char option()
{   char choice;
    printf("\n\n>> Enter your choice: ");
    switch(choice=getche())
    {
        case '1':
        case '2':
        case '3':
```

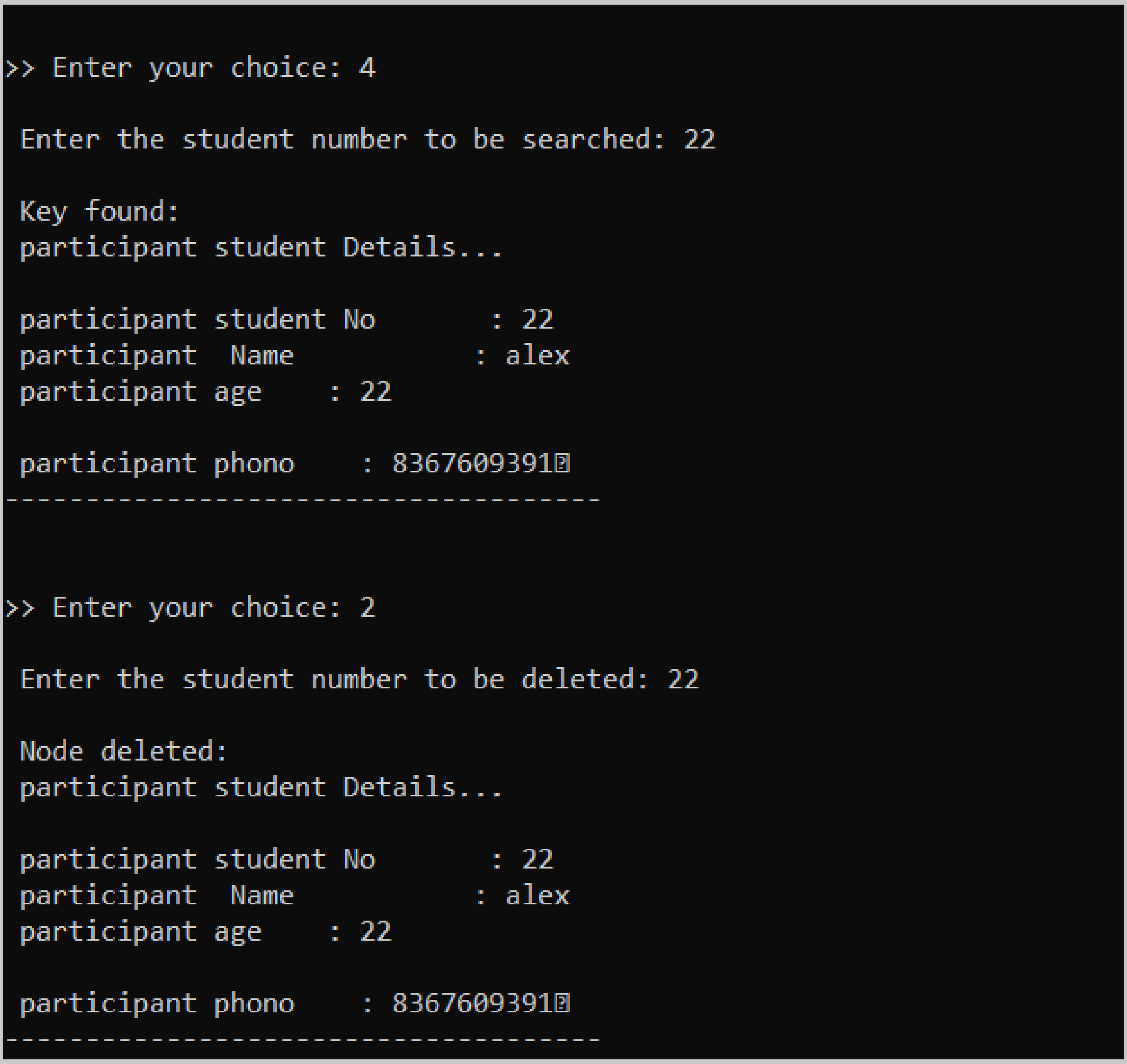
```
        case '4':
        case '5':   return(choice);
        default :   printf("\n Invalid choice.");
    }
    return choice;
}
void main()
{
    struct stud_data *linkList;
    char name[21];
    char choice;
    int rno;
    int age;
    int phono;
    linkList = NULL;
    printf("\n Welcome to demonstration of singly linked list \n");
    menu();
    do
    {
        choice = option();
        switch(choice)
        {
        case '1':
            printf("\n Enter the studentNumber      : ");
            scanf("%d", &rno);
            printf("Enter the student name        : ");
            fflush(stdin);
            gets(name);
            printf("Enter the student age : ");
            scanf("%f",&age);
            printf("\nEnter the student pho nunmber : ");
            scanf("%d",&phono);
            linkList = insert(linkList, rno, name, age,phono);
            break;
        case '2':
            printf("\n\n Enter the student number to be deleted: ");
            scanf("%d", &rno);
            linkList = deleteNode(linkList, rno);
            break;
        case '3':
            if (linkList == NULL)
            {
                printf("\n List empty.");
                break;
            }
            display(linkList);
            break;
        case '4':
            printf("\n\n Enter the student number to be searched: ");
            scanf("%d", &rno);
            search(linkList, rno);
            break;
        case '5':
            printf("\n\n Enter the student phone number   to be searched: ");
            scanf("%d", &phono);
            search(linkList, phono);
            break;
        case '6': break;
        }
    } while (choice != '6');
}
```

Out put:



"C:\Users\hp\Desktop\all subject\assessment3\bin\Debug\assessment3.exe"

```
 Welcome to demonstration of singly linked list
---------------------------------------------
Press 1 to INSERT a node into the list
Press 2 to DELETE a node from the list
Press 3 to DISPLAY the list
Press 4 to SEARCH the list
Press 5 to exit
---------------------------------------------



>> Enter your choice: 1
 Enter the studentNumber      : 22
Enter the student name        : alex

Enter the student pho nunmber : 8367609391
Enter the student age : 22


>> Enter your choice: 3
 participant student Details...

 participant student No       : 22
 participant  Name           : alex
 participant age     : 22

 participant phono    : 8367609391
-------------------------------------
```



```
>> Enter your choice: 4

 Enter the student number to be searched: 22

 Key found:
 participant student Details...

 participant student No       : 22
 participant  Name           : alex
 participant age    : 22

 participant phono    : 8367609391
-------------------------------------


>> Enter your choice: 2

 Enter the student number to be deleted: 22

 Node deleted:
 participant student Details...

 participant student No       : 22
 participant  Name           : alex
 participant age    : 22

 participant phono    : 8367609391
-------------------------------------
```

2,Create a Double Linked List of Student Details (Name, Register Number, and Year as parameters) who want to register for a certain course which has 10 slots. If a First year student opt to register the course, his details should be added to the end of the list (ie. First Year Students details must be in the end of the list if they opt), if Other year students' try to register, his details should be added in the list just before the first Year students'(if it exists). Go on getting 12 students' details and then display first 10 students' details who are admitted in the course.

Code:

```
#include<stdio.h>

#include<stdlib.h>

struct node

{

        char ssn[25],name[25],dept[10],designation[25];

        int year;

        char phone[10];

        struct node *llink;

        struct node *rlink;

};

typedef struct node* NODE;

NODE first = NULL;

int count=0;

NODE create()

{

         NODE enode;

         enode = (NODE)malloc(sizeof(struct node));

         if( enode== NULL)

         {

                  printf("\nRunning out of memory");

                  exit(0);

          }

          printf("\nEnter the ssn,Name,Department,Designation,phone,years of the student: \n");

        scanf("%s %s %s %s %s %d", enode->ssn, enode->name, enode->dept, enode-
>designation,enode->phone,&enode->year);

          enode->llink=NULL;

          enode->rlink=NULL;

          count++;

          return enode;

}
```

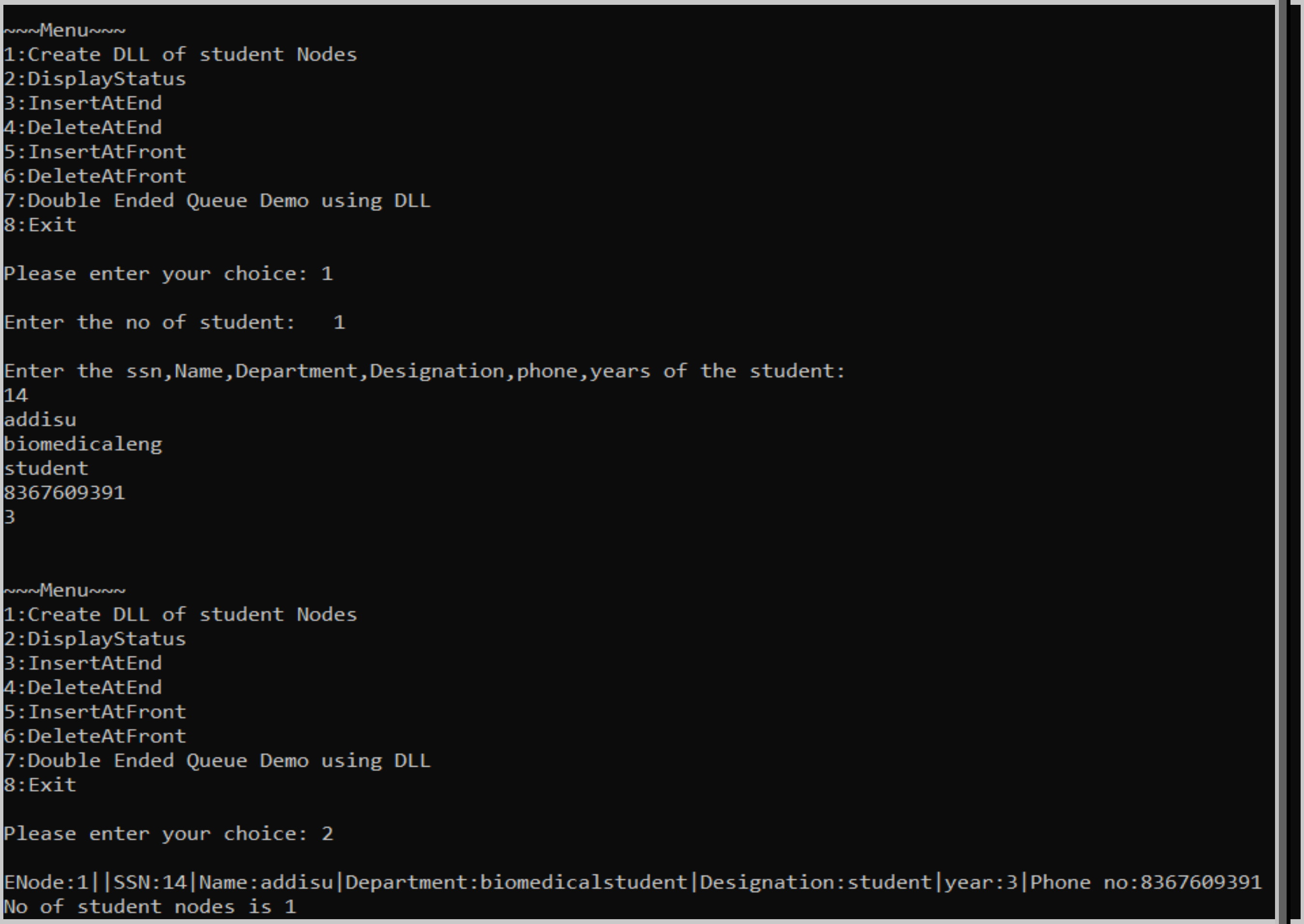
```
NODE insertfront()
{
        NODE temp;
        temp = create();
         if(first == NULL)
        {
               return temp;
         }
         temp->rlink = first;
         first->llink = temp;
         return temp;
}

void display()
{
         NODE cur;
         int nodeno=1;
         cur = first;
         if(cur == NULL)
                    printf("\nNo Contents to display in DLL");
         while(cur!=NULL)
        {
       printf("\nENode:%d||SSN:%s|Name:%s|Department:%s|Designation:%s|year:%d|Phone
no:%s", nodeno, cur->ssn, cur->name,cur->dept, cur->designation, cur->year, cur->phone);
                cur = cur->rlink;
                 nodeno++;
        }
         printf("\nNo of student nodes is %d",count);
}
```

```
NODE deletefront()
{
        NODE temp;
        if(first == NULL)
        {
                printf("\nDoubly Linked List is empty");
                return NULL;
        }
        if(first->rlink== NULL)
        {
               printf("\nThe student node with the ssn:%s is deleted", first->ssn);
              free(first);
              count--;
               return NULL;
        }
        temp = first;
        first = first->rlink;
        temp->rlink = NULL;
        first->llink = NULL;
        printf("\nThe student node with the ssn:%s is deleted",temp->ssn);
        free(temp);
        count--;
        return first;
}

NODE insertend()
{
         NODE cur, temp;
         temp = create();

         if(first == NULL)
         {
                   return temp;
         }
        cur= first;
        while(cur->rlink!=NULL)
       {
                 cur = cur->rlink;
        }
        cur->rlink = temp;
        temp->llink = cur;
         return first;
}
NODE deleteend()
{
        NODE prev,cur;
        if(first == NULL)
       {
               printf("\nDoubly Linked List is empty");
               return NULL;
       }
       if(first->rlink == NULL)
       {
                 printf("\nThe student node with the ssn:%s is deleted",first->ssn);
                 free(first);
                 count--;
                 return NULL;
       }

        prev=NULL;
        cur=first;
       while(cur->rlink!=NULL)
       {
                 prev=cur;
                 cur = cur->rlink;
        }
        cur->llink = NULL;
        printf("\nThe student node with the ssn:%s is deleted",cur->ssn);
        free(cur);
        prev->rlink = NULL;
        count--;
        return first;
}
void deqdemo()
{
      int ch;
      while(1)
     {
            printf("\nDemo Double Ended Queue Operation");
     printf("\n1:InsertQueueFront\n 2: DeleteQueueFront\n 3:InsertQueueRear\n
4:DeleteQueueRear\n 5:DisplayStatus\n 6: Exit \n");
            scanf("%d", &ch);


            switch(ch)
           {
                  case 1: first=insertfront();
                             break;
                   case 2: first=deletefront();
                             break;
                 case 3: first=insertend();
                             break;
                 case 4: first=deleteend();
```

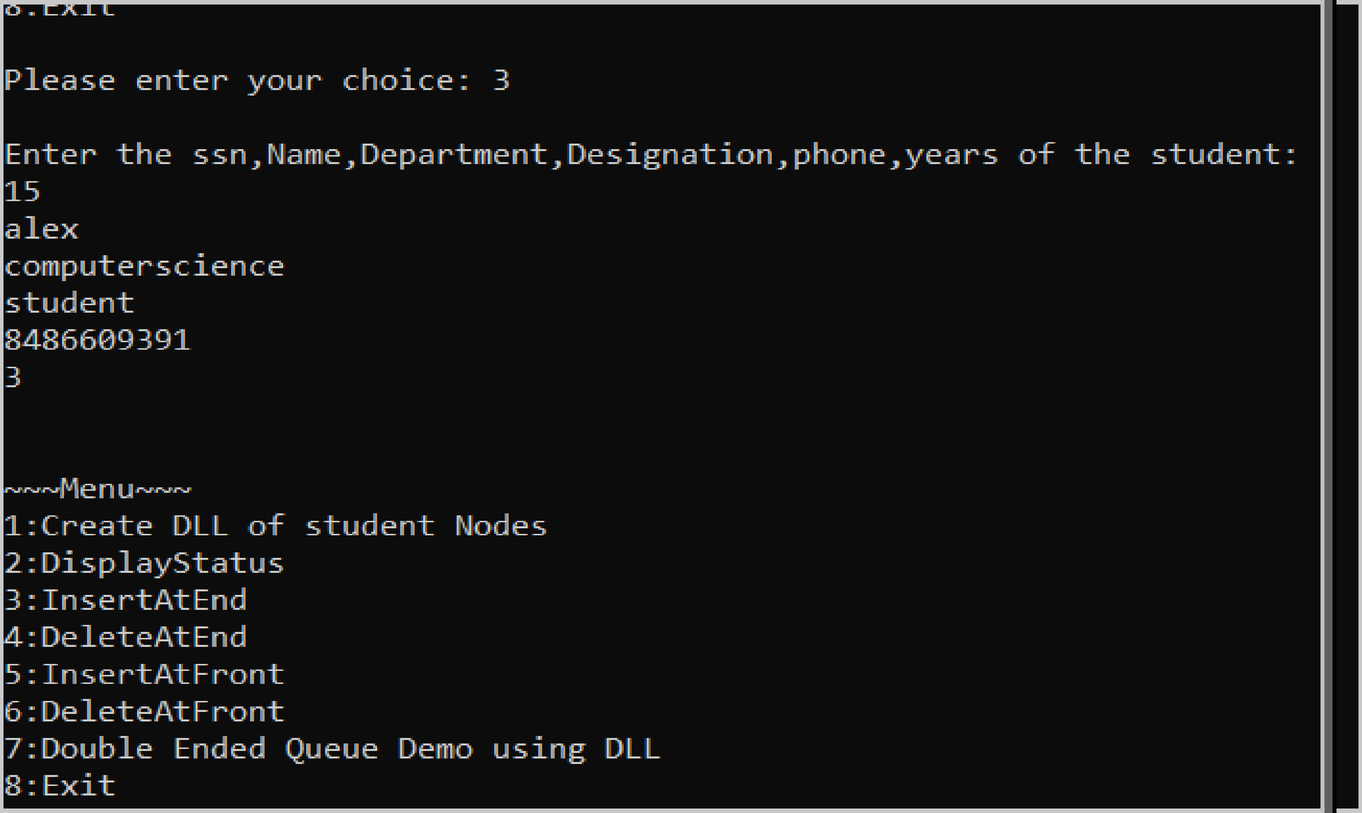
```
                        break;
                  case 5: display();
                        break;
                 default : return;
           }
     }
}

void main()
{
   int ch,i,n;
   while(1)
   {
       printf("\n\n~~~Menu~~~");
       printf("\n1:Create DLL of student Nodes");
       printf("\n2:DisplayStatus");
        printf("\n3:InsertAtEnd");
        printf("\n4:DeleteAtEnd");
        printf("\n5:InsertAtFront");
        printf("\n6:DeleteAtFront");
        printf("\n7:Double Ended Queue Demo using DLL");
        printf("\n8:Exit \n");
        printf("\nPlease enter your choice: ");
       scanf("%d",&ch);

       switch(ch)
       {
        case 1 : printf("\nEnter the no of student:   ");
                 scanf("%d",&n);
                 for(i=1;i<=n;i++)
                 first = insertend();
```

```
                 break;
         case 2:  display();
                 break;
          case 3: first = insertend();
                break;
          case 4: first = deleteend();
                break;
          case 5: first = insertfront();
                break;
          case 6: first = deletefront();
              break;
          case 7: deqdemo();
                break;
          case 8 : exit(0);
        default: printf("\nPlease Enter the valid choice");
        }
    }
}
```

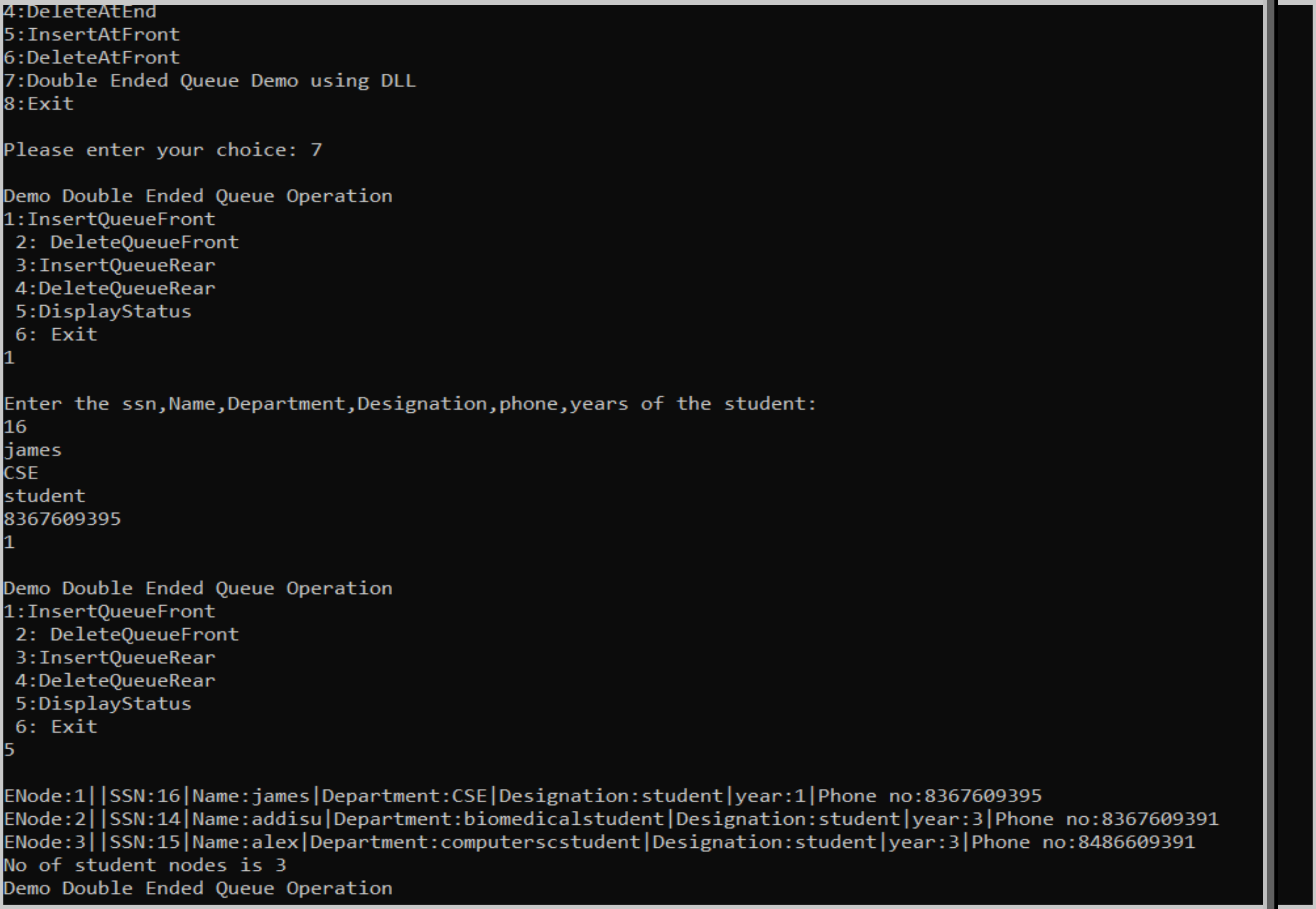
Output:



```
~~~Menu~~~
1:Create DLL of student Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
5:InsertAtFront
6:DeleteAtFront
7:Double Ended Queue Demo using DLL
8:Exit

Please enter your choice: 1

Enter the no of student:   1

Enter the ssn,Name,Department,Designation,phone,years of the student:
14
addisu
biomedicaleng
student
8367609391
3


~~~Menu~~~
1:Create DLL of student Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
5:InsertAtFront
6:DeleteAtFront
7:Double Ended Queue Demo using DLL
8:Exit

Please enter your choice: 2

ENode:1||SSN:14|Name:addisu|Department:biomedicalstudent|Designation:student|year:3|Phone no:8367609391
No of student nodes is 1
```



```
8:Exit

Please enter your choice: 3

Enter the ssn,Name,Department,Designation,phone,years of the student:
15
alex
computerscience
student
8486609391
3


~~~Menu~~~
1:Create DLL of student Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
5:InsertAtFront
6:DeleteAtFront
7:Double Ended Queue Demo using DLL
8:Exit
```

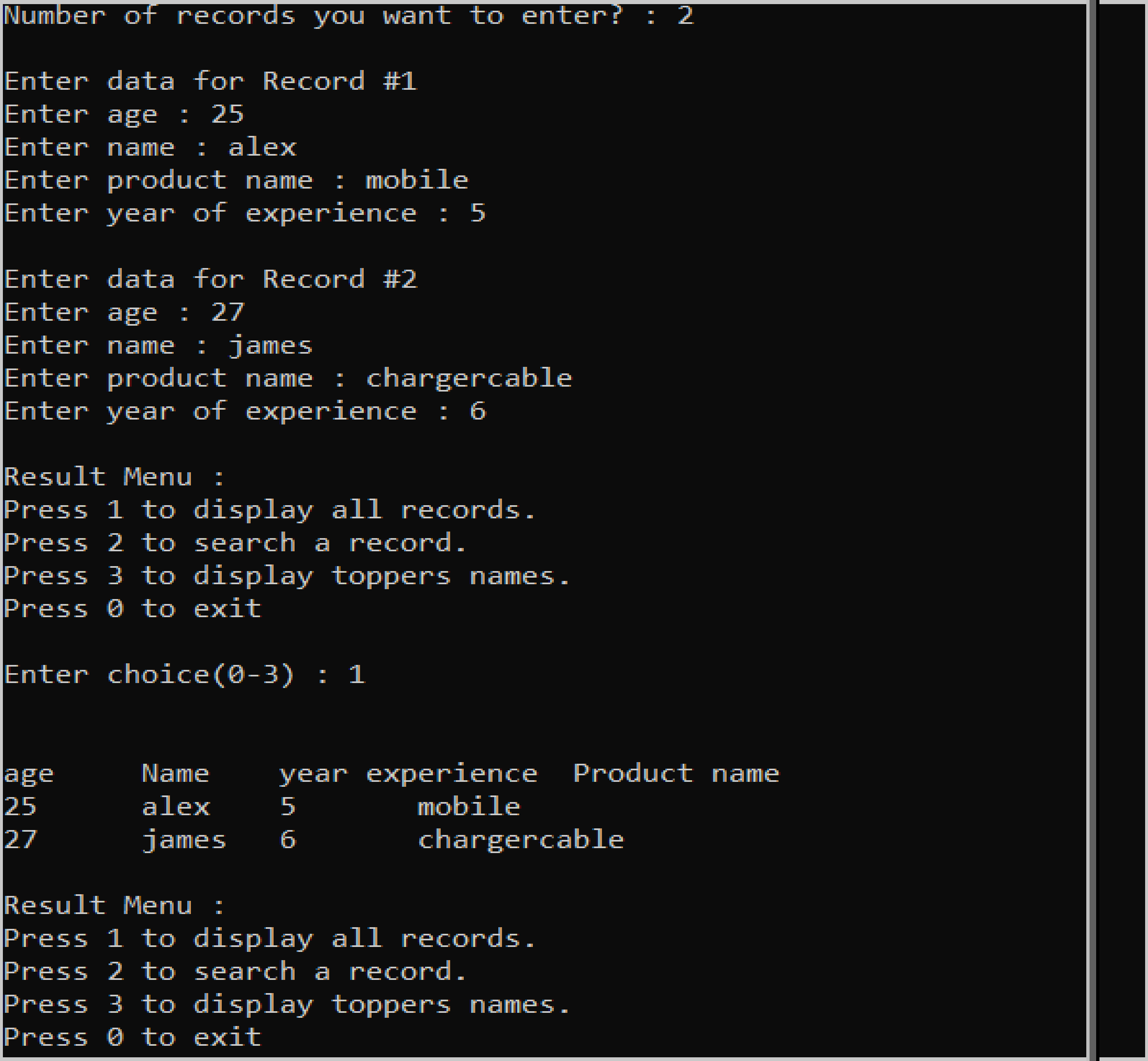

```
4:DeleteAtEnd
5:InsertAtFront
6:DeleteAtFront
7:Double Ended Queue Demo using DLL
8:Exit

Please enter your choice: 7

Demo Double Ended Queue Operation
1:InsertQueueFront
 2: DeleteQueueFront
 3:InsertQueueRear
 4:DeleteQueueRear
 5:DisplayStatus
 6: Exit
1

Enter the ssn,Name,Department,Designation,phone,years of the student:
16
james
CSE
student
8367609395
1

Demo Double Ended Queue Operation
1:InsertQueueFront
 2: DeleteQueueFront
 3:InsertQueueRear
 4:DeleteQueueRear
 5:DisplayStatus
 6: Exit
5

ENode:1||SSN:16|Name:james|Department:CSE|Designation:student|year:1|Phone no:8367609395
ENode:2||SSN:14|Name:addisu|Department:biomedicalstudent|Designation:student|year:3|Phone no:8367609391
ENode:3||SSN:15|Name:alex|Department:computerscstudent|Designation:student|year:3|Phone no:8486609391
No of student nodes is 3
Demo Double Ended Queue Operation
```

3 Write a C program that takes the details of sales representatives (name, age, products and years of experience) .Sort the data based on years of experience using bubble sort. Display the details of the agents who represent a pair of products in common.

Code:

```
#include <stdio.h>

struct sale_representative

{

   int age;

   char name[80];

   int yearexp;

   char product[80];

};

void accept(struct sale_representative[], int);

void display(struct sale_representative[], int);
void search(struct sale_representative[], int, int);
int findMax(struct sale_representative[], int);
void toppers(struct sale_representative[], int);
int main()
{
   struct sale_representative data[20];
   int n, choice, age;
   printf("Number of records you want to enter? : ");
   scanf("%d", &n);
   accept(data, n);
   do
   {
      printf("\nResult Menu :\n");
      printf("Press 1 to display all records.\n");
      printf("Press 2 to search a record.\n");
      printf("Press 3 to display toppers names.\n");
      printf("Press 0 to exit\n");
      printf("\nEnter choice(0-3) : ");
      scanf("%d", &choice);
      switch (choice)
      {
         case 1:
            display(data, n);
            break;
         case 2:
            printf("Enter age number to search : ");
            scanf("%d", &age);
            search(data, n, age);
            break;
            case 3:
        toppers(data, n);
        }
    }
    while (choice != 0);
    return 0;
}
void accept(struct sale_representative list[80], int s)
{
    int i;
    for (i = 0; i < s; i++)
    {
        printf("\nEnter data for Record #%d", i + 1);
        printf("\nEnter age : ");
        scanf("%d", &list[i].age);
        fflush(stdin);
        printf("Enter name : ");
        gets(list[i].name);
         fflush(stdin);
        printf("Enter product name : ");
        gets(list[i].product);
        printf("Enter year of experience : ");
        scanf("%d", &list[i].yearexp);
    }
}
void display(struct sale_representative list[80], int s)
{
    int i;
    printf("\n\nage\tName\tyear experience\t Product name\n");
    for (i = 0; i < s; i++)
    {
        printf("%d\t%s\t%d\t%s\n", list[i].age, list[i].name, list[i].yearexp,list[i].product);
    }
}
void search(struct sale_representative list[80], int s, int number)
{
    int i;
    for (i = 0; i < s; i++)
    {
        if (list[i].age== number)
        {
            printf("age: %d\nName : %s\nyear experience : %d\n", list[i].age,
                list[i].name, list[i].yearexp);
            return ;
        }
    }
    printf("Record not Found\n");
}
int findMax(struct sale_representative list[], int s)
{
    int i, max;
    max = list[0].yearexp;
    for (i = 1; i < s; i++)
    {
        if (list[i].yearexp > max)
        {
            max = list[i].yearexp;
        }
    }
return max;
}
void toppers(struct sale_representative list[], int s)
{
    int i;
    for (i = 0; i < s; i++)
    {if (list[i].yearexp== findMax(list, s))
        {
            printf("%s\n", list[i].name);
        }
    }
}
```

Out put:


```
Number of records you want to enter? : 2

Enter data for Record #1
Enter age : 25
Enter name : alex
Enter product name : mobile
Enter year of experience : 5

Enter data for Record #2
Enter age : 27
Enter name : james
Enter product name : chargercable
Enter year of experience : 6

Result Menu :
Press 1 to display all records.
Press 2 to search a record.
Press 3 to display toppers names.
Press 0 to exit

Enter choice(0-3) : 1


age     Name    year experience  Product name
25      alex    5       mobile
27      james   6       chargercable

Result Menu :
Press 1 to display all records.
Press 2 to search a record.
Press 3 to display toppers names.
Press 0 to exit
```

4, Write a C++ program to perform the following operations on binary search tree of strings: {Dhanush, Bala, Elumalai, Arun, Bhuvanesh, Himanshu, Garima, Indrajit, Faisal, James} • Create a binary search tree • Insert the following strings into a binary search tree {Harish,Ajay} • Delete the following strings from above binary search tree{ Bhuvanesh,Arun, Indrajit, Himanshu} • Search for a string in a binary search tree {Ajay,Harish}.

C++ code:

```
#include <iostream>
```

```
#include <cstdlib>
#include <string>
using namespace std;
struct binarySearch { //CREATING OUR BINARY NODE
string info;
binarySearch *Left, *Right; //CHILD FOR THE NODE
};
binarySearch* root; //GLOBAL HEAD OF NODE
class Binary_tree {
public:
Binary_tree();
void insert(string);
binarySearch* insertIntoTree(binarySearch*, binarySearch*);
void Delete(string);
void pretrav(binarySearch*);
void intrav(binarySearch*);
void posttrav(binarySearch*);
};
Binary_tree::Binary_tree()
{
root = NULL;
}
binarySearch* Binary_tree::insertIntoTree(binarySearch* temp, binarySearch* newnode)
{
if (temp == NULL) {
temp = newnode;
}
else if (temp->info < newnode->info) { //DOING STRING COMPARISION FOR THE INSERTION
insertIntoTree(temp->Right, newnode);
if (temp->Right == NULL)
temp->Right = newnode;
```

```
}
else {
insertIntoTree(temp->Left, newnode);
if (temp->Left == NULL)
temp->Left = newnode;
}
return temp;
}
void Binary_tree::insert(string n) //INSERTION TO CHECK IF IT'S FIRST NODE
{
binarySearch *temp = root, *newnode;
newnode = new binarySearch;
newnode->Left = NULL;
newnode->Right = NULL;
newnode->info = n;
root = insertIntoTree(temp, newnode);
}
void Binary_tree::pretrav(binarySearch* t = root)
{
if (root == NULL) {
cout << "Nothing to display";
}
else if (t != NULL) {
cout << t->info << " ";
pretrav(t->Left);
pretrav(t->Right);
}
}
void Binary_tree::intrav(binarySearch* t = root)//INLINE TRAVESING
{
if (root == NULL) {
```

```
cout << "Nothing to display";
}
else if (t != NULL) {
intrav(t->Left);
cout << t->info << " ";
intrav(t->Right);
}
}
void Binary_tree::posttrav(binarySearch* t = root)//POST TRAVESING
{
if (root == NULL) {
cout << "Nothing to display";
}
else if (t != NULL) {
posttrav(t->Left);
posttrav(t->Right);
cout << t->info << " ";
}
}
void Binary_tree::Delete(string key) //DELETE FROM THE TREE
{
binarySearch *temp = root, *parent = root, *marker;
if (temp == NULL)
cout << "The tree is empty" << endl;
else {
while (temp != NULL && temp->info != key) {
parent = temp;
if (temp->info < key) {
temp = temp->Right;
}
else {
```

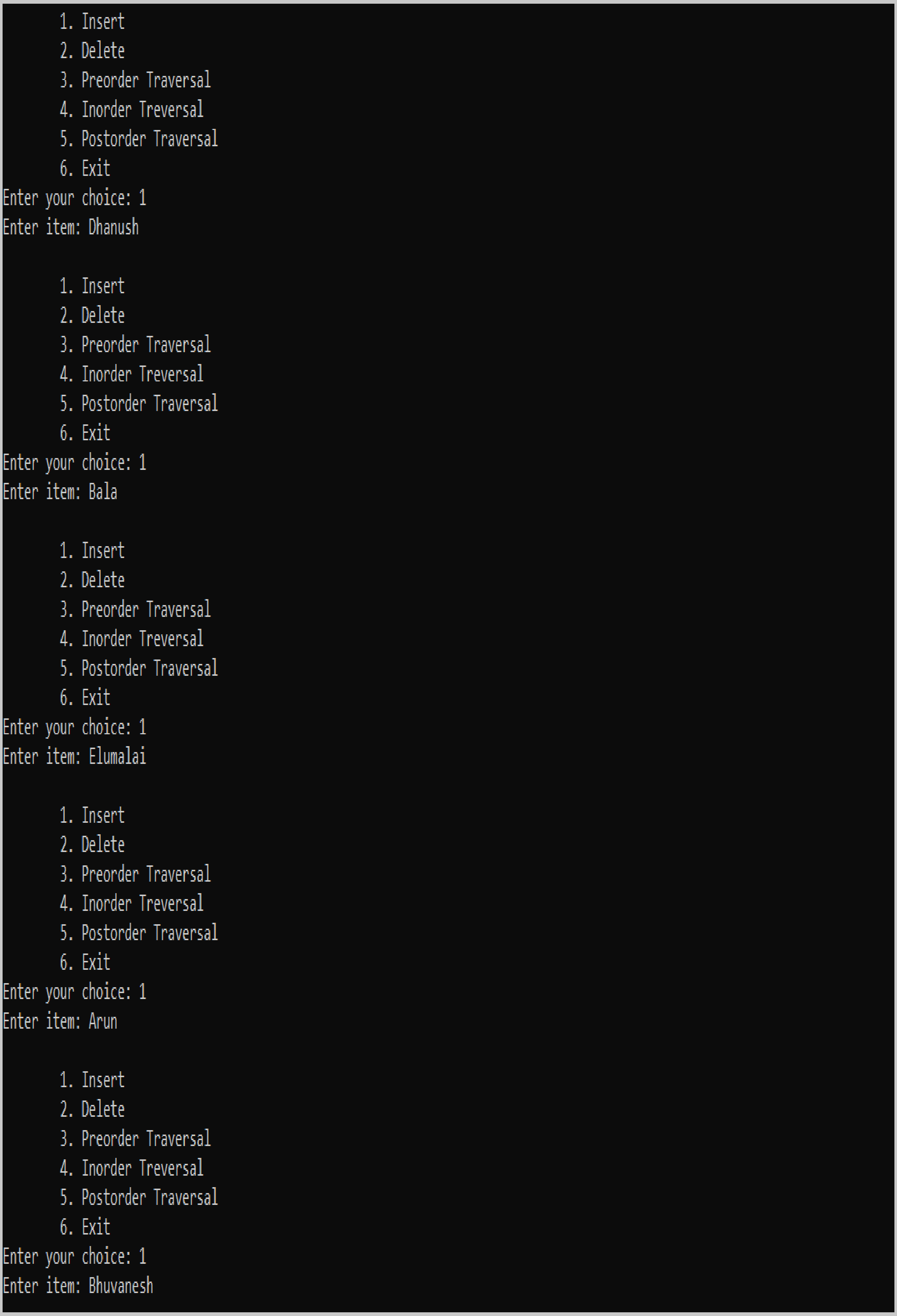
```
temp = temp->Left;}
}
}
marker = temp;
if (temp == NULL)
cout << "No node present";
else if (temp == root) {
if (temp->Right == NULL && temp->Left == NULL) {
root = NULL;
}
else if (temp->Left == NULL) {
root = temp->Right;
}
else if (temp->Right == NULL) {
root = temp->Left;
}
else {
binarySearch* temp1;
temp1 = temp->Right;
while (temp1->Left != NULL) {
temp = temp1;
temp1 = temp1->Left;
}
if (temp1 != temp->Right) {
temp->Left = temp1->Right;
temp1->Right = root->Right;
}
temp1->Left = root->Left;
root = temp1;
}
}
```

```
else {
if (temp->Right == NULL && temp->Left == NULL) {
if (parent->Right == temp)
parent->Right = NULL;
else
parent->Left = NULL;
}
else if (temp->Left == NULL) {
if (parent->Right == temp)
parent->Right = temp->Right;
else
parent->Left = temp->Right;
}
else if (temp->Right == NULL) {
if (parent->Right == temp)
parent->Right = temp->Left;
else
parent->Left = temp->Left;
}
else {
binarySearch* temp1;
parent = temp;
temp1 = temp->Right;
while (temp1->Left != NULL) {
parent = temp1;
temp1 = temp1->Left;
}
if (temp1 != temp->Right) {
temp->Left = temp1->Right;
temp1->Right = parent->Right;
}
```

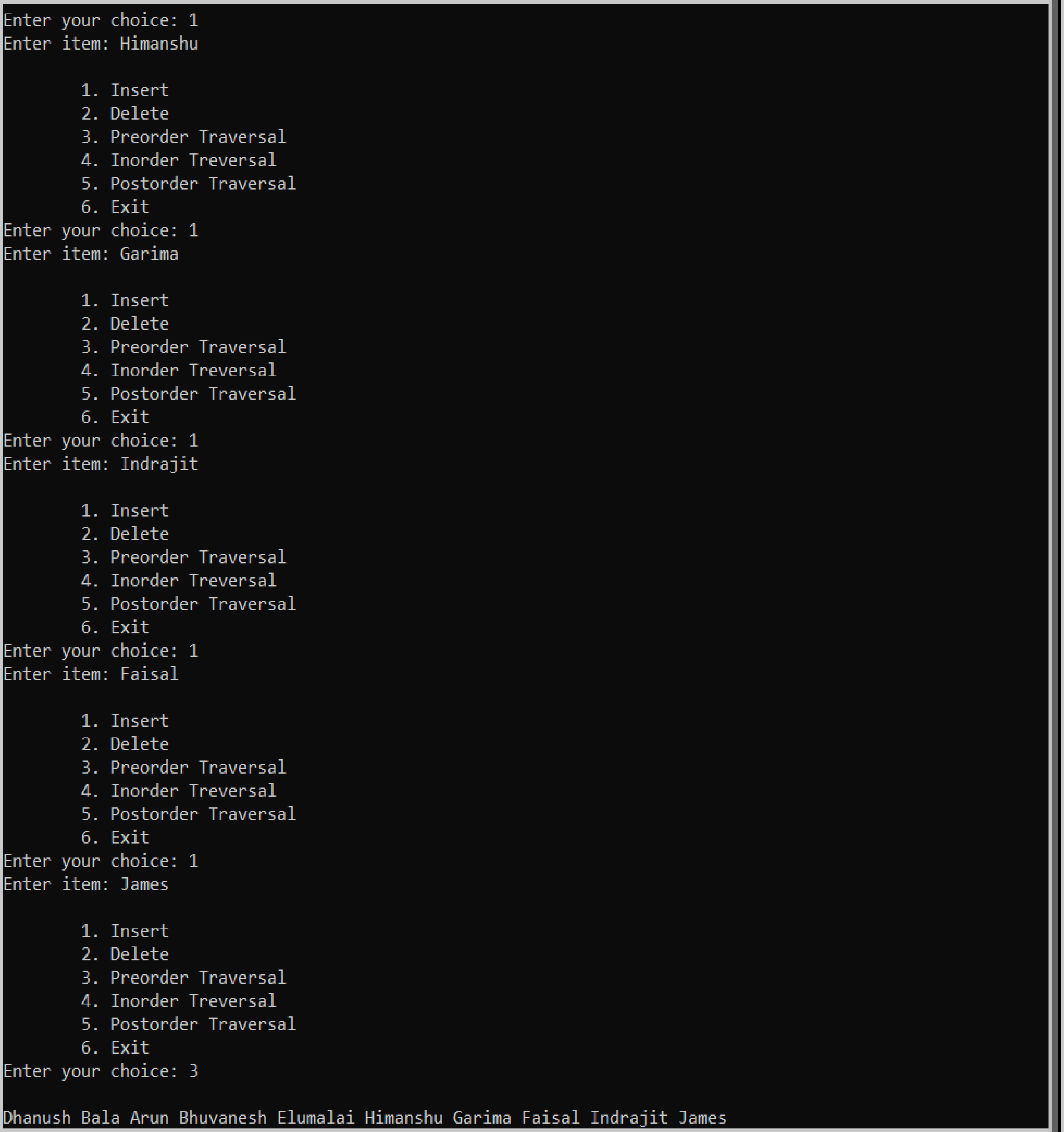
```
temp1->Left = parent->Left;

parent = temp1;

}

}

delete marker;

}

int main()

{

Binary_tree bt;

string n, key;

int choice;

while (1) {

cout << "\n\t1. Insert\n\t2. Delete\n\t3. Preorder Traversal\n\t4. Inorder Treversal\n\t5. Postorder
Traversal\n\t6. Exit" << endl;

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

cout << "Enter item: ";

cin >> n;

bt.insert(n);

break;

case 2:

cout << "Enter element to delete: ";

cin >> key;

bt.Delete(key);

break;

case 3:

cout << endl;

bt.pretrav();

break;
```

```
case 4:
cout << endl;
bt.intrav();
break;
case 5:
cout << endl;
bt.posttrav();
break;
case 6:
exit(0);
}
}
return 0;
}
```

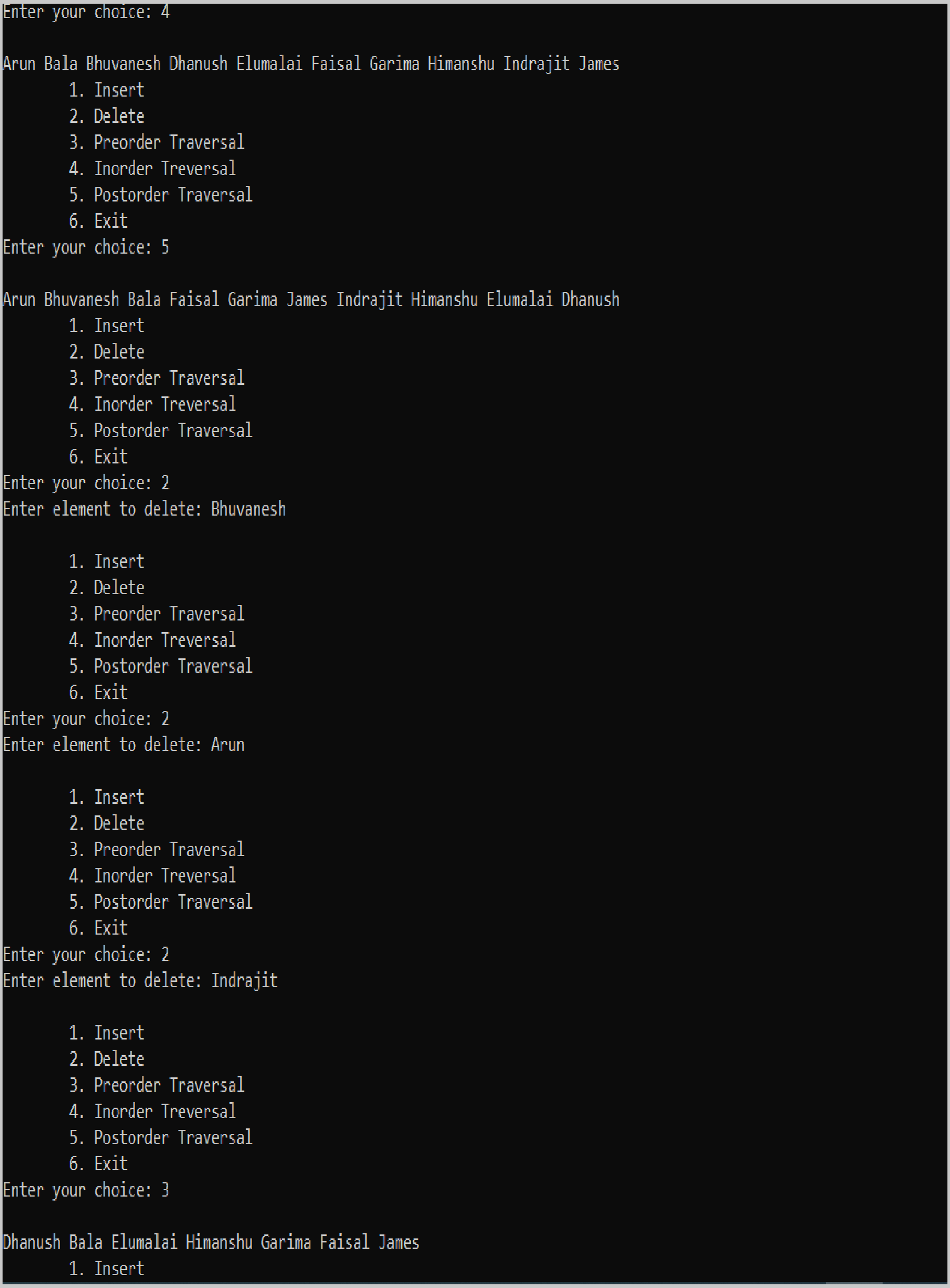
Output:



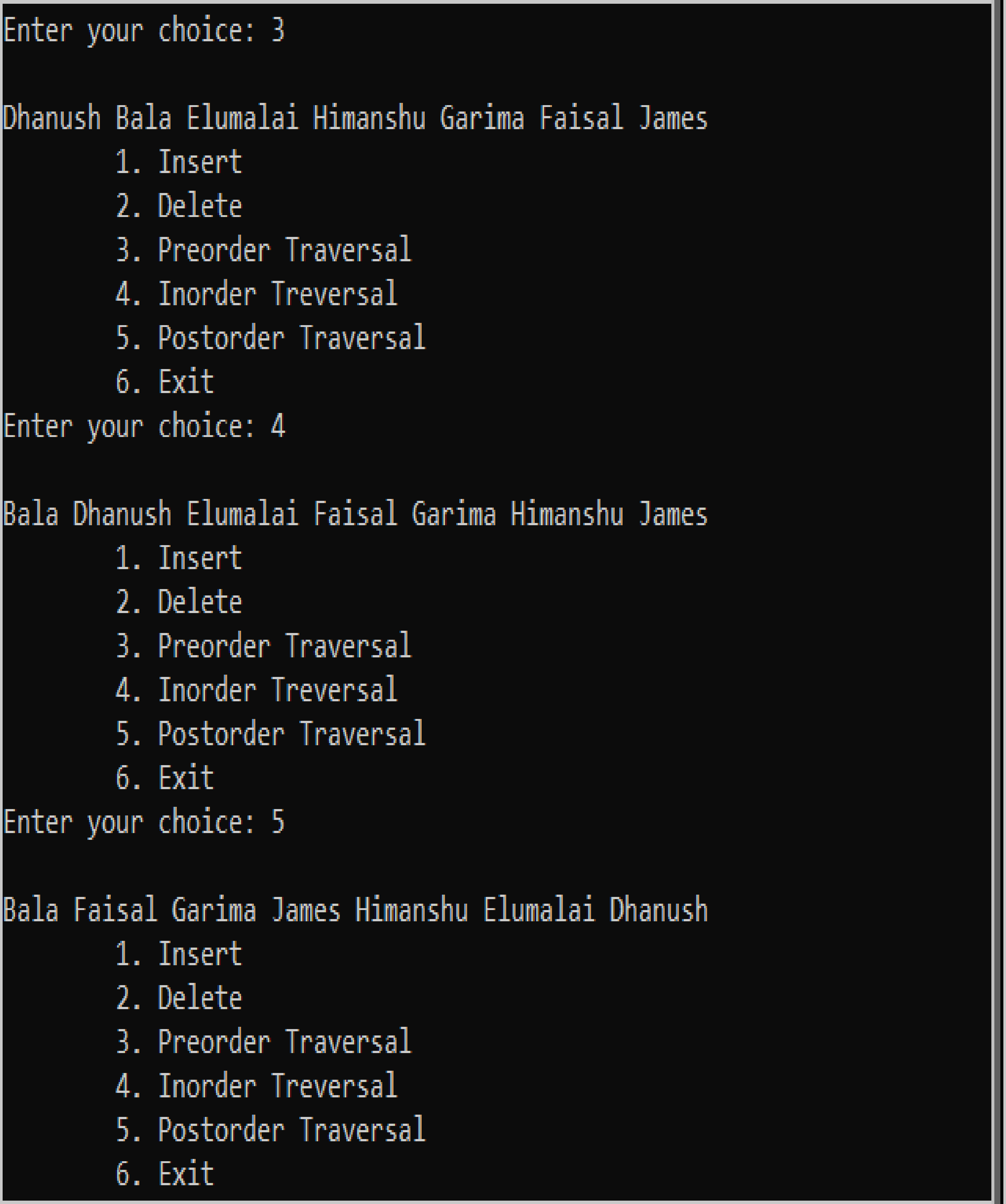
```
        1. Insert
        2. Delete
        3. Preorder Traversal
        4. Inorder Treversal
        5. Postorder Traversal
        6. Exit
Enter your choice: 1
Enter item: Dhanush

        1. Insert
        2. Delete
        3. Preorder Traversal
        4. Inorder Treversal
        5. Postorder Traversal
        6. Exit
Enter your choice: 1
Enter item: Bala

        1. Insert
        2. Delete
        3. Preorder Traversal
        4. Inorder Treversal
        5. Postorder Traversal
        6. Exit
Enter your choice: 1
Enter item: Elumalai

        1. Insert
        2. Delete
        3. Preorder Traversal
        4. Inorder Treversal
        5. Postorder Traversal
        6. Exit
Enter your choice: 1
Enter item: Arun

        1. Insert
        2. Delete
        3. Preorder Traversal
        4. Inorder Treversal
        5. Postorder Traversal
        6. Exit
Enter your choice: 1
Enter item: Bhuvanesh
```



```
Enter your choice: 1
Enter item: Himanshu

        1. Insert
        2. Delete
        3. Preorder Traversal
        4. Inorder Treversal
        5. Postorder Traversal
        6. Exit
Enter your choice: 1
Enter item: Garima

        1. Insert
        2. Delete
        3. Preorder Traversal
        4. Inorder Treversal
        5. Postorder Traversal
        6. Exit
Enter your choice: 1
Enter item: Indrajit

        1. Insert
        2. Delete
        3. Preorder Traversal
        4. Inorder Treversal
        5. Postorder Traversal
        6. Exit
Enter your choice: 1
Enter item: Faisal

        1. Insert
        2. Delete
        3. Preorder Traversal
        4. Inorder Treversal
        5. Postorder Traversal
        6. Exit
Enter your choice: 1
Enter item: James

        1. Insert
        2. Delete
        3. Preorder Traversal
        4. Inorder Treversal
        5. Postorder Traversal
        6. Exit
Enter your choice: 3

Dhanush Bala Arun Bhuvanesh Elumalai Himanshu Garima Faisal Indrajit James
```



```
Enter your choice: 4

Arun Bala Bhuvanesh Dhanush Elumalai Faisal Garima Himanshu Indrajit James
        1. Insert
        2. Delete
        3. Preorder Traversal
        4. Inorder Treversal
        5. Postorder Traversal
        6. Exit
Enter your choice: 5

Arun Bhuvanesh Bala Faisal Garima James Indrajit Himanshu Elumalai Dhanush
        1. Insert
        2. Delete
        3. Preorder Traversal
        4. Inorder Treversal
        5. Postorder Traversal
        6. Exit
Enter your choice: 2
Enter element to delete: Bhuvanesh

        1. Insert
        2. Delete
        3. Preorder Traversal
        4. Inorder Treversal
        5. Postorder Traversal
        6. Exit
Enter your choice: 2
Enter element to delete: Arun

        1. Insert
        2. Delete
        3. Preorder Traversal
        4. Inorder Treversal
        5. Postorder Traversal
        6. Exit
Enter your choice: 2
Enter element to delete: Indrajit

        1. Insert
        2. Delete
        3. Preorder Traversal
        4. Inorder Treversal
        5. Postorder Traversal
        6. Exit
Enter your choice: 3

Dhanush Bala Elumalai Himanshu Garima Faisal James
        1. Insert
```



```
Enter your choice: 3

Dhanush Bala Elumalai Himanshu Garima Faisal James
        1. Insert
        2. Delete
        3. Preorder Traversal
        4. Inorder Treversal
        5. Postorder Traversal
        6. Exit
Enter your choice: 4

Bala Dhanush Elumalai Faisal Garima Himanshu James
        1. Insert
        2. Delete
        3. Preorder Traversal
        4. Inorder Treversal
        5. Postorder Traversal
        6. Exit
Enter your choice: 5

Bala Faisal Garima James Himanshu Elumalai Dhanush
        1. Insert
        2. Delete
        3. Preorder Traversal
        4. Inorder Treversal
        5. Postorder Traversal
        6. Exit
```

5,

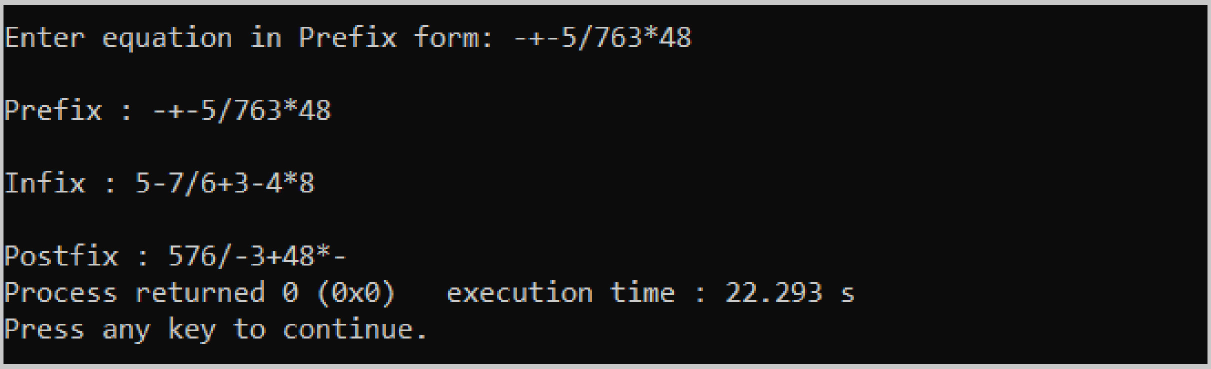
Construct an expression tree for the following prefix expression. Find the corresponding postfix expression by traversing the tree in post order. Evaluate that postfix expression using stacks. - + - 5 / 7 6 3 * 4 8

Code:

```
#include <iostream>

#include <cstdlib>

#include <cstdio>

#include <cstring>

using namespace std;

class TreeN{//node declaration {

  public:

  char d;

  TreeN *l, *r;

  TreeN(char d) {

    this->d = d;

    this->l = NULL;

    this->r = NULL;

  }

};

class StackNod// stack declaration {

{

 public: TreeN *treeN;;

  StackNod *n;

  StackNod(TreeN*treeN)//constructor

   {

    this->treeN = treeN;

    n = NULL;

  }

};

class ExpressionTree {
```

```
private: StackNod *top;
public: ExpressionTree() {
  top = NULL;
}
void clear() {
  top = NULL;
}
void push(TreeN *ptr) {
  if (top == NULL)
    top = new StackNod(ptr);
  else {
    StackNod *nptr = new StackNod(ptr);
    nptr->n = top;
    top = nptr;
  }
}

TreeN *pop() {
  if (top == NULL) {
    cout<<"Underflow"<<endl;
  } else {
    TreeN *ptr = top->treeN;
    top = top->n;
    return ptr;
  }
}
TreeN *peek() {
  return top->treeN;
}
void insert(char val) {
  if (isDigit(val)) {
    TreeN *nptr = new TreeN(val);
    push(nptr);
  } else if (isOperator(val)) {
    TreeN *nptr = new TreeN(val);
    nptr->l = pop();
    nptr->r= pop();
    push(nptr);
  } else {
    cout<<"Invalid Expression"<<endl;
    return;
  }
}

bool isDigit(char ch) {
  return ch >= '0' && ch <= '9';
}
bool isOperator(char ch) {
  return ch == '+' || ch == '-' || ch == '*' || ch == '/';
}
int toDigit(char ch) {
  return ch - '0';
}
void buildTree(string eqn) {
  for (int i = eqn.length() - 1; i >= 0; i--)
    insert(eqn[i]);
}

void postfix() {
  postOrder(peek());
}
void postOrder(TreeN*ptr) {
  if (ptr != NULL) {
    postOrder(ptr->l);
    postOrder(ptr->r);
    cout<<ptr->d;
  }
}
void infix() {
  inOrder(peek());
}

void inOrder(TreeN *ptr) {
  if (ptr != NULL) {
    inOrder(ptr->l);
    cout<<ptr->d;
    inOrder(ptr->r);
  }
}
void prefix() {
  preOrder(peek());
}
void preOrder(TreeN *ptr) {
  if (ptr != NULL) {
    cout<<ptr->d;
    preOrder(ptr->l);
    preOrder(ptr->r);
  }
}
};
int main() {
string s;
ExpressionTree et;

cout<<"\nEnter equation in Prefix form: ";

cin>>s;

et.buildTree(s);

cout<<"\nPrefix : ";

et.prefix();

cout<<"\n\nInfix : ";

et.infix();

cout<<"\n\nPostfix : ";

et.postfix();

}
```

Out put:



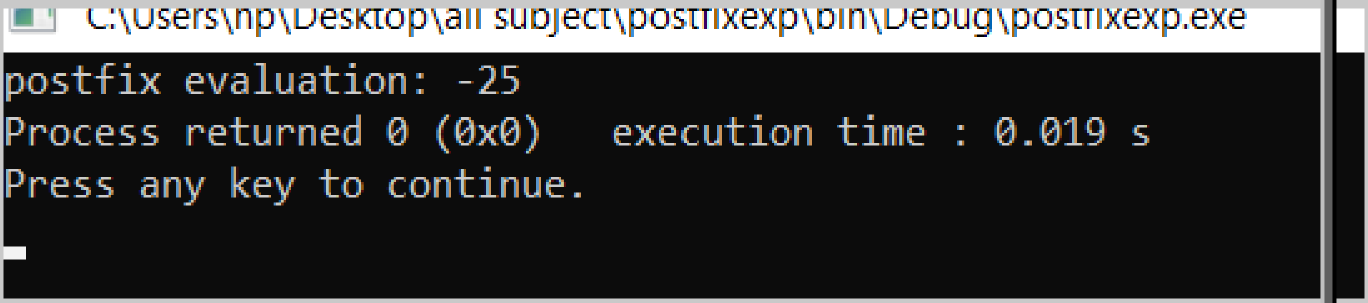
```
Enter equation in Prefix form: -+-5/763*48

Prefix : -+-5/763*48

Infix : 5-7/6+3-4*8

Postfix : 576/-3+48*-
Process returned 0 (0x0)   execution time : 22.293 s
Press any key to continue.
```

Code for evaluation of postfix expression using stack:

```
#include <stdio.h>

#include <string.h>

#include <ctype.h>

#include <stdlib.h>

struct Stack

{

   int top;

   unsigned capacity;

   int* array;

};
```

```
struct Stack* createStack( unsigned capacity )
{
    struct Stack* stack = (struct Stack*) malloc(sizeof(struct Stack));
    if (!stack) return NULL;
    stack->top = -1;
    stack->capacity = capacity;
    stack->array = (int*) malloc(stack->capacity * sizeof(int));
    if (!stack->array) return NULL;
    return stack;
}
int isEmpty(struct Stack* stack)
{
    return stack->top == -1 ;
}
char peek(struct Stack* stack)
{
    return stack->array[stack->top];
}
char pop(struct Stack* stack)
{
    if (!isEmpty(stack))
        return stack->array[stack->top--] ;
    return '$';
}
void push(struct Stack* stack, char op)
{
    stack->array[++stack->top] = op;
}
int evaluatePostfix(char* exp)
{
    struct Stack* stack = createStack(strlen(exp));
    int i;
    if (!stack) return -1;
    for (i = 0; exp[i]; ++i)
    {
        if (isdigit(exp[i]))
            push(stack, exp[i] - '0');
        else
        {
            int val1 = pop(stack);
            int val2 = pop(stack);
            switch (exp[i])
            {
            case '+': push(stack, val2 + val1); break;
            case '-': push(stack, val2 - val1); break;
            case '*': push(stack, val2 * val1); break;
            case '/': push(stack, val2/val1); break;
            }
        }
    }
    return pop(stack);
}
int main()
{
    char exp[] = "576/-3+48*-";
    printf ("postfix evaluation: %d", evaluatePostfix(exp));
    return 0;
}
```

Output:


```
C:\Users\hp\Desktop\all subject\postfixexp\bin\Debug\postfixexp.exe
postfix evaluation: -25
Process returned 0 (0x0)   execution time : 0.019 s
Press any key to continue.
```

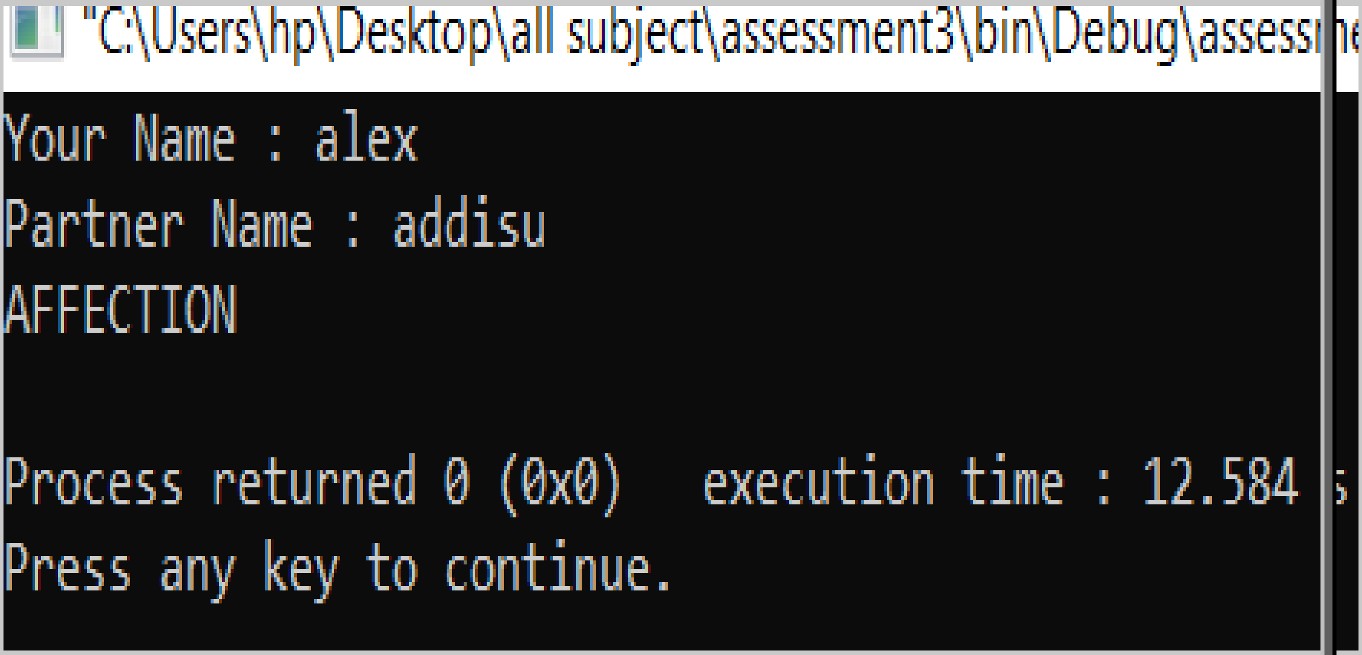
6. Assume FLAMES game that tests for relationship has to be implemented using a dynamic structure. The letters in the FLAMES stand for Friends, Love, Affection, Marriage, Enmity and Sister. Initially store the individual letters of the word ‘flames’ in the nodes of the dynamic structure. Given the count of the number of uncommon letters in the two names ‘n’, write a program to delete every nth node in it, till it is left with a single node. If the end of the dynamic structure is reached while counting, resume the counting from the beginning. Display the letter that still remains and the corresponding relationship Eg., If Ajay and Jack are the two names, there are 4 uncommon letters in these. So delete 4th node in the first iteration and for the next iteration start counting from the node following the deleted node. Low Level: Delete only the first nth node only Middle Level: Implement the above problem to delete every nth node till the list is left with a single node. High Level: For the same problem instead of deleting the nth node, make the nth node as the last node. Hence at last the first node gives the relationship.

Code:

```
#include<stdio.h>

#include<string.h>

char* Flames( char fl[],int n,int fcount)

{

    int j=0,i,k;

    char buff[10]="";        //Initializing Buffer to 0

    if(strlen(fl)==1){

       return fl;         //Last Remaining character in fl[] returned

    }

    else

    {

        for(i=0;i<fcount;i++)       //Traverse through fl[]="flames"

       {

         if(j==n)

            j=0;      // if j reaches last character then j initialize to start 0

         j++;
```

```
            }
            fl[j-1]='\0';       //Put NULL to Cancel the character in fl[]
            k=0;

            /*make copy of remaining character in fl[] char array.
             Eg. if fl[]="fl\0mes" it will store in buff as start with j to n-1 times
             buff[]="mesfl\0"*/

            for(i=j;k<n;i++,k++){
               if(i==n)
                   i=0;
               buff[k]=fl[i];
            }
            Flames(buff,strlen(buff),fcount); //Do this process recursively until fl[] becomes single
character
        }
}
int flamesCount(char* first,char* second)
{
    int i,j;
    int len=strlen(first)+strlen(second);
    for(i=0;i<strlen(second);i++)
        if(second[i]==32)      //Neglecting White spaces of Partner name
            len--;
    for(i=0;i<strlen(first);i++) {
        if(first[i]!=32) {
            for(j=0;j<strlen(second);j++)
                    if(second[j]!=32)
                        if(first[i]==second[j]) {
                            len=len-2;       //If Two Characters are same then minus 2 with overall length
                            first[i]=32;
                        second[j]=32;break;
                      }
        }
        else
          len--;      //Neglecting White spaces of Your name
      }
      return len;
    }
    int main()
    {
      char first[20];
      char second[20];
      char fl[]="flames";
      char* ans;
      printf("Your Name : ");
      scanf("%[^\n]%*c",first);
      printf("Partner Name : ");
      scanf("%[^\n]%*c",second);
      int Fcount=flamesCount(first,second);
      if(Fcount==0){
        printf("AFFECTION\n");  //Two Names are SAME then Print Affection in Default
        return 0;}
      ans=Flames(fl,6,Fcount);
      switch(ans[0])
      {
        case 'f' : printf("FRIENDS\n");break;
        case 'l' : printf("LOVERS\n");break;
        case 'a' : printf("AFFECTION\n");break;
        case 'm' : printf("MARRIAGE\n");break;
        case 'e' : printf("ENEMIES\n");break;
        case 's' : printf("SIBLINGS\n");break;
        default  : printf("Error");break;
    }
    return 0;
}
```

Out put:



```
"C:\Users\hp\Desktop\all subject\assessment3\bin\Debug\assessm
Your Name : alex
Partner Name : addisu
AFFECTION


Process returned 0 (0x0)   execution time : 12.584 s
Press any key to continue.
```