|  |  |
| --- | --- |
| **Ex.no.1a** | **Implementation of symbol table using hashing** |

## Head.h

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

typedef struct node{

char symbol[20];

int length;

struct node \*next;

}hashtable;

struct hash{

struct node \*head;

};

struct hash \*array;

int insert(hashtable,hashtable \*);

hashtable\* findprev(char[],hashtable \*);

hashtable\* find(char[],hashtable \*);

void display(hashtable \*);

int modify(hashtable,hashtable \*);

int del(char[],hashtable\*);

## Implementation.c

#include"head.h"

int insert(hashtable h,hashtable \*start){

hashtable \*temp;

temp=(hashtable\*)malloc(sizeof(hashtable));

if(temp==NULL){

printf("Memory not allocated\n");

return 0;

}

strcpy(temp->symbol,h.symbol);

temp->length=h.length;

hashtable \*p=findprev(temp->symbol,start);

temp->next=p->next;

p->next=temp;

return 1;

}

hashtable\* findprev(char tempsymbol[],hashtable \*start){

hashtable \*temp=start->next;

hashtable \*prev=start;

while(temp!=NULL){

if(strcmp(temp->symbol,tempsymbol)>0)

return prev;

prev =temp;

temp=temp->next;

}

return prev;

}

hashtable\* find(char tempsymbol[],hashtable \*start){

hashtable \*temp=start->next;

while(temp!=NULL){

if(strcmp(temp->symbol,tempsymbol)==0)

return temp;

temp=temp->next;

}

return NULL;

}

void display(hashtable \*p){

if(p == NULL)

printf("\nnot found\n");

else

printf("\t%s\t%d\n",p->symbol,p->length);

}

int modify(hashtable h,hashtable \*start){

hashtable \*temp=start->next;

if((temp=find(h.symbol,start))==NULL)

return 0;

temp->length=h.length;

return 1;

}

int del(char tempsymbol[] , hashtable \*start){

hashtable \*temp=start->next;

hashtable \*prev=start;

if(find(tempsymbol,start)==NULL)

return 0;

while(temp!=NULL){

if(strcmp(temp->symbol,tempsymbol)==0)

break;

prev=temp;

temp=temp->next;

}

prev->next=temp->next;

free(temp);

return 1;

}

## Application.c

#include "head.h"

int main(){

int ch,i;

char tsymb[10];

struct hash h[25];

hashtable temp;

for(i=0;i<25;i++){

hashtable \*temp;

temp=(hashtable\*)malloc(sizeof(hashtable));

if(temp==NULL){

printf("Hash table not created\n");

exit(0);

}

temp->next=NULL;

strcpy(temp->symbol,"\0");

h[i].head=temp;

}

while(1){

printf("1-Insert\n2-Find\n3-Modify\n4-Delete\n5-Exit\n");

printf("Enter Your Choice : ");

scanf("%d",&ch);

switch(ch){

case 1: printf("Enter the Symbol : ");

scanf("%s",temp.symbol);

printf("Enter the Length : ");

scanf("%d",&temp.length);

if(insert(temp,h[temp.symbol[0]%65].head))

printf("inserted\n");

else

printf("\ninsertion failed\n");

break;

case 2: printf("Enter symbol : ");

scanf("%s",tsymb);

display(find(temp.symbol,h[tsymb[0]%65].head));

break;

case 3: printf("Enter Symbol : ");

scanf("%s",temp.symbol);

printf("Enter the new Length : ");

scanf("%d",&temp.length);

if(modify(temp,h[temp.symbol[0]%65].head)){

printf("modified\n");

display(find(temp.symbol,h[tsymb[0]%65].head));

}

else

printf("modification failed\n");

break;

case 4: printf("\nEnter a Symbol to Delete : ");

scanf("%s",tsymb);

if(del(tsymb,h[tsymb[0]%65].head))

printf("Deleted\n");

else

printf("deletion failed");

break;

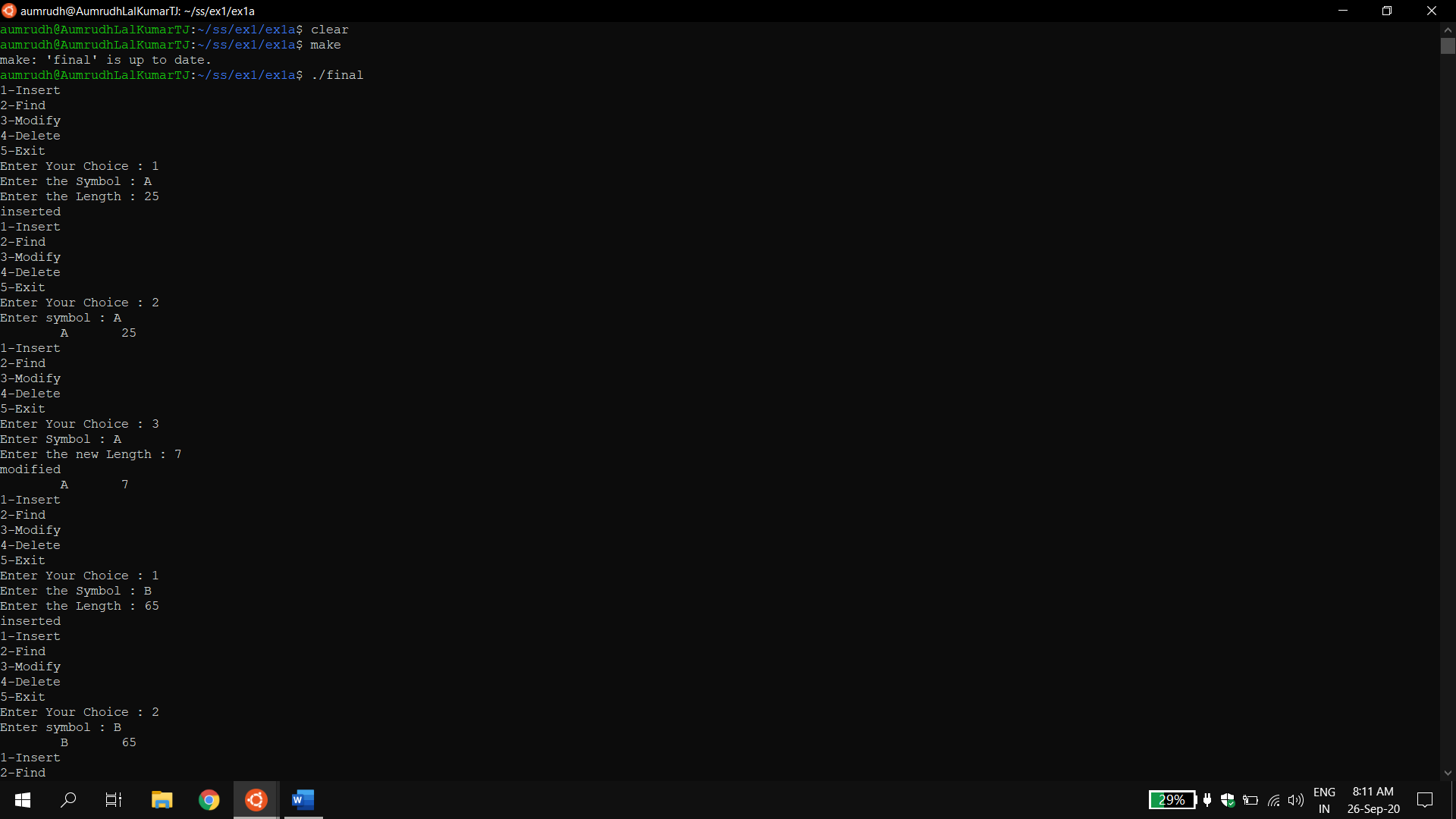
case 5: exit(1);

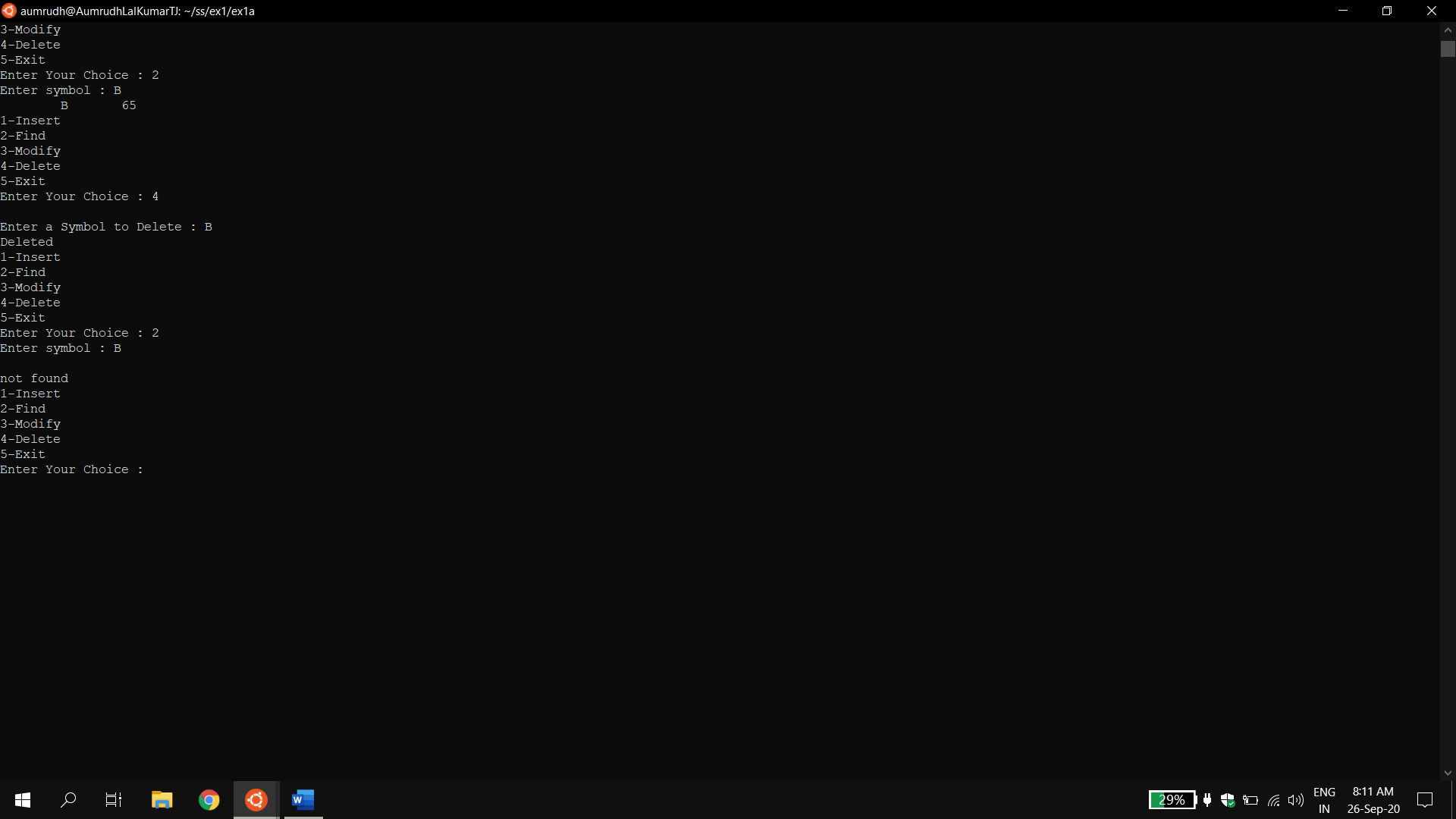
}

}

}

## Output:





## Result:

The symtab implementation using hash table for insert, search, modify and delete operation was done successfully.