Rajshahi University of Engineering & Technology

Department of Computer Science of Engineering

**Experiment No**: 05,06 & 07

**Name of Experiment**: Two Way Linked Lists,Stack,Recursion & Queue

|  |
| --- |
| Submitted to:  Rizoan Toufiq  Assistant Professor  Department of Computer Science & Engineering  Rajshahi University of Engineering & Technology |

|  |
| --- |
| Submitted by:  Name: Md. Ariful Islam  Roll No.: 1803046  Group: 2nd Thirty  Date of Exp.: 02-10-2019  Date of sub. : 09-11-2019  Series: 18 Series |

Machine configuration:

ASUS X510UF

CORE I5 8th Gen Processor

Up To 3.4 GHz

8 GB RAM

OS WIN 10

INDEX

|  |  |  |
| --- | --- | --- |
| Preblem No. | Name of Problem | Page No. |
| 01. | Creating a two way linked list | 04-06 |
| 02. | Adding an item into a stack (PUSH) | 07-08 |
| 03. | Deleting an item from a stack (POP) | 09-10 |
| 04. | Calculating factorial of n with recursion | 10 |
| 05. | Calculating nth term of fibonacci sequence with recursion | 11 |
| 06. | Adding an item into a queue (PUSH) | 12-13 |
| 07. | Deleting an item from a queue (POP) | 14-15 |
| 08 | <empty> |  |
| 09 | <empty> |  |
| 10. | <empty> |  |

# Theory: A two way linked list is a linear collection of data elements,called **nodes**,where the linear order is given by means of pointers. Each node is divided into three parts :

# The first part contains the **information** of the elements.

# The second part called the **Forward field** or **next pointer field** that contains the address of the next node in the list.

# The third part called the **Backward field** or **previous pointer field** that contains the address of the previous node in the list.

# The operations normally performed on any linear structures are :

1. Traversal : Processing each elements in the list.
2. Search : Finding the location of an element.
3. Insertion : Adding a new element to the list.
4. Deletion : Removing an element from the list.

Etc.

A Stack is a linear structure in which items may be added or removed one by one only at the end. It means that the last item to be added to a stack is the first item to be removed. The main concept of stack is Last In - First Out.

# In a stack there is a pointer **TOP** that shows the **location** of **last data item.** There are two operations that are normally performed on any stack :

1. PUSH : Inserting an element into a stack.
2. POP : Deleting an element from a stack.

The process in which a function calls itself directly or indirectly is called recursion and the corresponding function is called as recursive function.

A Queue is a linear structure in which items may be added one by one only at the end or removed one by one only at the beginning.It means that the last item to be added to a queue is the last item to be removed.

The main concept of queue is First In - First Out.

# In a queue there is a pointer **REAR** that shows the **location** of **last data item.** There are two operations that are normally performed on any queue :

1. PUSH : Inserting an element into a queue.
2. POP : Deleting an element from a queue.

**Problem 1:** Creating a two way linked list.

**Flow Chart:**

START

START : NULL

START : NULL

INFO[NEW] : ITEM

FORW[NEW] : NULL

i : 1

i++

i : n

>

<

yes

AVAIL : NULL

no

END

NEW : AVAIL

AVAIL : FORW[AVAIL]

START : NEW

PTR : START

BACK[NEW]: NULL

= =

!=

LINK[PTR] : NEW

PTR : NEW

BACK[FORW[PTR]]: PTR

**Algorithm:**

# (Creating a Two Way Linked List) This algorithm create a linked list with n nodes.

# START := NULL

# Repeat Steps 3 to 5 for I = 1 to N

# [OVERFLOW] If AVAIL = NULL, then:

# Write: OVERFLOW, and Exit.

# [Remove first node from AVAIL.]

# Set NEW:=AVAIL and AVAIL := FORW[AVAIL]

# Set INFO[NEW] := ITEM and FORW[NEW] := NULL

# If START = NULL, then:

# Set START := NEW and PTR : = START and BACK[NEW]:=NULL

# Else:

# Set FORW[PTR] := NEW and PTR = NEW and BACK[FORW[PTR]]:=PTR

# [End of If structure]

1. Exit.

Code:

|  |
| --- |
| #include<stdio.h>  char info[15];  int forw[10]={2,3,4,5,6,7,8,9,1,-1};  int back[10];  int start=-1;  int avail=0;  static int end=-1;  int newnode(){  int newindex;  if(avail==-1){  printf("\nOverflow\n"); return -1;  }  else{  newindex=avail;  avail=forw[avail];  return newindex;  }  }  void creat\_list(){  int ptr=-1,i,newindex;  char ch='H';  for(i=0;i<10;i++){  newindex=newnode();  if(newindex==-1)  break;  info[newindex]=ch;  forw[newindex]=-1;  if(start==-1){  start=newindex;  back[newindex]=-1;  ptr=newindex;  end=newindex;  }  else{  forw[ptr]=newindex;  back[forw[ptr]]=ptr;  ptr=newindex;  end=newindex;  }  ch++;  }  ptr=-1; }  int main(){  creat\_list();  traverse\_list();  return 0;  } |

Problem No: 02

Problem Statement: Adding an item into a stack (PUSH).

Flow Chart :

START

AVAIL : NULL

no yes

NEW: AVAIL

AVAIL: LINK[AVAIL]

LINK[NEW]: NULL

OVERFLOW

INFO[NEW] : ITEM

LINK[TOP]: NEW

TOP: NEW

END

# 

# **Algorithm:** **PUSH(INFO,LINK, TOP, AVAIL,NEW, ITEM)**

# This procedure pushes an ITEM into a stack.

# [Stack already filled]

# IF AVAIL = NULL, then Write: OVERFLOW, and Return

# Set NEW:= AVAIL and AVAIL:= LINK[AVAIL]

LINK[NEW]:= NULL

# 3. Set INFO[NEW]:= ITEM and LINK[TOP]:= NEW and TOP:= NEW.

# 4. Return.

Code:

|  |
| --- |
| #include<stdio.h>  int main(){  int info[10]={10,20,30,40,50,0,0,0,0,0};  int link[10]={2,3,4,5,0,7,8,9,10,0};  int start=1,top=5,avail=6,ptr,New,item;    if(avail==0)  printf("Overflow\n");  else{  New=avail;  avail=link[avail-1];  link[New-1]=0;  scanf("%d",&item);  info[New-1]=item;  link[top-1]=New;  }  return 0;  } |

Problem No: 03

Problem Statement: Deleting an item from a stack (POP).

Flow Chart :

START

TOP : NULL

no yes

ITEM: INFO[TOP]

LINK[TOP]: AVAIL

AVAIL: TOP

UNDERFLOW

TOP : TOP-1

LINK[TOP]: NULL

END

# 

# **Algorithm:** **POP(INFO,LINK, TOP, AVAIL,NEW, ITEM)**

# This procedure pushes an ITEM into a stack.

# [Stack already filled]

# IF AVAIL = NULL, then Write: UNDERFLOW, and Return

# Set ITEM:= INFO[TOP] and LINK[TOP] and AVAIL:= TOP.

# 3. Set TOP:= TOP-1 and LINK[TOP]:= NULL.

# 4. Return.

Code:

|  |
| --- |
| #include<stdio.h>  int main(){  int info[10]={10,20,30,40,50,0,0,0,0,0};  int link[10]={2,3,4,5,0,7,8,9,10,0};  int start=1,top=5,avail=6,ptr,New,item;    if(top==0)  printf("Underflow\n");  else{  item=info[top-1];  link[top-1]=avail;  avail=top;  top=top-1;  link[top-1]=0;  }  return 0;  } |

Problem No: 04

Problem Statement: Calculating factorial of n with recursion.

# **Algorithm: FACTORIAL(FACT, N)**

# This process calculates N! And returns the value in the variable FACT.

1. If N:=0, then : Set FACT:= 1 and Return.
2. Call FACTORIAL(FACT,N-1).
3. Set FACT:= FACT\*N.
4. Return.

Code:

|  |
| --- |
| #include<stdio.h>  int factorial(int fact,int n){  if(n==0)  fact=1;  else  fact=n\*factorial(fact,n-1);  return fact;  }  int main(){  int n,fact;  scanf("%d",&n);  fact=factorial(fact,n);  printf("\nFactorial %d = %d\n",n,fact);  return 0;  } |

Problem No: 05

Problem Statement: Calculating nth term of fibonacci sequence with recursion.

# **Algorithm: FIBONACCI(FIB, N)**

# This process calculates FN and returns the value in the first parameter FIB.

1. If N:=0 or N:= 1, then : Set FIB:= N and Return.
2. Call FIBONACCI(FIBA, N-2).
3. Call FIBONACCI(FIBB, N-1).
4. Set FIB:= FIBA+FIBB.
5. Return.

Code:

|  |
| --- |
| #include<stdio.h>  int fibonacci(fib,n){  int fiba,fibb;  if(n==0||n==1)  fib=n;  else  fib=fibonacci(fib,n-2)+fibonacci(fib,n-1);  return fib;  }  int main(){  int n,fib;  scanf("%d",&n);  fib=fibonacci(fib,n);  printf("\n\n%dth of Fibonacci Sequence = %d\n",n,fib);  } |

Problem No: 06

Problem Statement: Adding an item into a queue (PUSH).

Flow Chart :

START

AVAIL : NULL

no yes

NEW: AVAIL

AVAIL: LINK[AVAIL]

LINK[NEW]: NULL

OVERFLOW

INFO[NEW] : ITEM

LINK[TOP]: NEW

TOP: NEW

END

# **Algorithm:** **PUSH(INFO,LINK,TOP,AVAIL, ITEM)**

# This procedure pushes an ITEM into a stack.

# IF AVAIL:= NULL, then Write: OVERFLOW, and Return.

# [Stack already filled]

# Else Set NEW:= AVAIL and AVAIL:= LINK[AVAIL]

# and LINK[NEW]:= NULL.

# Set INFO[NEW]:= ITEM and LINK[TOP]:=NEW

# and TOP:= NEW.

# 4. Return.

Code:

|  |
| --- |
| #include<stdio.h>  int main(){  int info[10]={10,20,30,40,50,0,0,0,0,0};  int link[10]={2,3,4,5,0,7,8,9,10,0};  int start=1,top=5,avail=6,ptr,New,item;  if(avail==0)  printf("Overflow\n");  else{  New=avail;  avail=link[avail-1];  link[New-1]=0;  scanf("%d",&item);  info[New-1]=item;  link[top-1]=New;  top=New;  }  return 0;  } |

Problem No: 07

Problem Statement: Deleting an item from a queue (POP).

Flow Chart :

START

TOP : NULL

no yes

ITEM: INFO[START]

NEW: LINK[START]

LINK[START]: AVAIL

UNDERFLOW

AVAIL: START

START: NEW

END

# **Algorithm:** **POP(INFO,LINK,TOP,AVAIL, ITEM)**

# This procedure pushes an ITEM into a stack.

# IF TOP:= NULL, then Write: UNDERFLOW, and Return.

# [Stack already empty]

# Else Set ITEM:= INFO[START] and NEW:= LINK[START]

# and LINKSTART]:= AVAIL.

# Set AVAIL:= START and START:=NEW.

# 4. Return.

Code:

|  |
| --- |
| #include<stdio.h>  int main(){  int info[10]={10,20,30,40,50,0,0,0,0,0};  int link[10]={2,3,4,5,0,7,8,9,10,0};  int start=1,top=5,avail=6,ptr,New,item;    if(top==0)  printf("Underflow\n");  else{  item=info[start-1];  New=link[start-1];  link[start-1]=avail;  avail=start;  start=New;  }  return 0;  } |

# THE END #